Report No. 433

COO-2118-0005

ILLIAC III REFERENCE MANUAL

VOLUME I : The Computer System

edited by
B. H. McCormick and B. J. Nordmann, Jr.

D. E. Atkins, R. T. Borovec, L. N. Goyal, L. M. Katoh,
R. M. Lansford, J. C. Schwebel and V. G. Tareski

February 17, 1971

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

COO-2118-0005

Report No. 433

ILLIAC III REFERENCE MANUAL

VOLUME I : The Computer System

edited by

B. H. McCormick and B. J. Nordmann, Jr.

D. E. Atkins, R. T. Borovec, L. N. Goyal, L. M. Katoh,
R. M. Lansford, J. C. Schwebel and V. G. Tareski

February 17, 1971

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

This work was supported by Contract AT (11-1)-1018 with the U. S. Atomic Energy Commission through September 30, 1970. Current support is under Contract AT (11-1)-2118 with the above agency.

ABSTRACT

The Illiac III Reference Manual is being issued in this final documentation as four volumes:

This issue---Volume I: The Computer System
Volume II: Instruction Repertoire
Volume III: Input/Output
Volume IV: Supervisor Organization

For ease of cross-reference an integrated table of contents will be issued separately.

This first volume reviews the overall system design of the Illinois Pattern Recognition Computer (Illiac III) and discusses the 14 subsystems comprising the computer system. In particular, all available documentation on each subsystem (taxicrinic processor, arithmetic unit, pattern articulation unit, etc.) is listed as of this date of issue.

ACKNOWLEDGMENTS

The Illiac III Reference Manual is based in part upon two earlier reports:

(1) B.H. McCormick (editor), William D. Bond, Kimio Ibuki, Roger E. Wiegel and John A. Wilber, Preliminary Programming Manual for the Illiac III Computer, Department of Computer Science Report 185, University of Illinois, July 1965.

(2) B.H. McCormick and R.M. Lansford (editors), D.E. Atkins, R.T. Borovec, G.N. Cederquist, S.K. Chan, L.A. Dunn, J.P. Hayes, L.M. Katoh, P.L. Koo, B.J. Nordmann, Jr., J.A. Rohr, and J.C. Schwebel, Illiac III Programming Manual, Department of Computer Science Manual, University of Illinois, March 1968.

The present editors (B.H. McCormick and B.J. Nordmann, Jr.) acknowledge with gratitude the contribution of these earlier groups as transplanted to this greatly expanded manual. In addition, Dr. Rangaswamy Narasimhan contributed significantly to the early definition of the pattern articulation unit instruction set. In like manner Philip Merryman assisted materially in the early formulation of machine-implemented macros--the imprimitive instructions of Illiac III.

Authors and contributors to the manual are listed by principal area of concern:

Taxicrinic Processor:	B.J. Nordmann, Jr., R.M. Lansford, J.C. Schwebel, R.E. Wiegel
Input/Output Processor:	L.M. Katoh, V.G. Tareski, J.V. Wenta, G.N. Cederquist, J.P. Hayes
Arithmetic Units:	D.E. Atkins, L.M. Goyal
Pattern Articulation Unit:	R.T. Borovec, R.P. Harms, G.T. Lewis
Interrupt Unit:	L.M. Goyal
Exchange Net:	S.K. Chan, P. Krabbe
Scanner-Monitor-Video	L.A. Dunn, L.M. Goyal, V.G. Tareski, R.G. Martin, R.C. Amendola
Supervisor Organization:	B.J. Nordmann, Jr., R.M. Lansford

The seemingly endless drafts and revisions of this manual have been handled with patience and fortitude by Mrs. Betty Gunsalus and Mrs. Roberta Andre'. Illustrations were prepared by Mr. Stanley Zundo.

ILLIAC III REFERENCE MANUAL

0. INTRODUCTION

This manual provides a functional description of the Illiac III Computer from a programmer's point of view. Each instruction of the machine is fully explained, noting error conditions, indicators which may be set, etc. Only detail necessary for programming is given in the description of the computer: features under program control, input/output device characteristics affecting programming, etc. Explanations of aspects of the system design which need not concern the programmer (such as wired-in selection of an available arithmetic unit, etc.) have been omitted.

The manual is divided into four main sections. The first volume gives an overall view of the Illiac III Computer System. The approach taken here presents the block diagram of the system (Figure 1): the interconnection of computer modules is first given and then each module is described. In the second volume, each module (input/output excepted) is reexamined; in particular the instruction set of each module is described in detail. The special class of imprimitive instructions is introduced here and explained. Then reflecting the traditional separation of input/output from computation, volume III is devoted exclusively to input/output. Volume IV then descrives in outline the Supervisor of the computer.

No specific knowledge of Illiac III is required prior to reading this manual, but previous experience in machine/assembly language programming is desirable, if not essential. For the programmer interested only in general purpose computation, most information will be found prior to Section 2.3 inclusive and in Section 3. The programmer interested in the Pattern Articulation Unit should also read Section 2.4. System programmers, particularly those associated with input/output programming, will need to be familiar with the entire manual.

1. THE ILLIAC III COMPUTER SYSTEM

The Illinois Pattern Recognition Computer, Illiac III, is a digital processor for visual information. It is primarily designed for automatic scanning and analysis of massive amounts of relatively homogeneous visual data. In particular the design is an outgrowth of studies at this laboratory of a computer system capable of scanning, measuring and analyzing in excess of 3×10^6 bubble chamber negatives per year.

The Illiac III, though specifically designed to process visual information, also provides complete facilities for standard, general-purpose computation. Both the picture processing and general-purpose computation facilities of the Illiac III will be available to users on a time-sharing basis.

As can be seen in Figure 1, Illiac III has been designed as a multi-processor computer system. Six processors (4 Taxicrinic Processors and 2 Input/Output Processors) can access in parallel the computational/storage units consisting of 2 Arithmetic Units, 1 Pattern Articulation Unit, 1 Interrupt Unit and 4 Storage Groups. Each computational/storage unit of the computer system specializes in a particular activity. Thus, for example, all floating-point computation is done in the Arithmetic Units, while picture processing is performed primarily by the Pattern Articulation Unit. Processors, on the other hand, analyze user jobs and route their constituent tasks to the appropriate specialized processing units. The individual processors of the system can operate simultaneously and independently (within the limits imposed by the Operating System Supervisor) with a consequent increase in overall efficiency.

The Input/Output Processors (IOP) are attached via Channel Interface Units and Device Controllers to various input and output devices. Among facilities important for the ingestion of visual infor-

mation are 7 CRT flying spot scanners: two for 70 mm film, two for 46 mm film, two for microfilm/microfiche, and one for microscope slides. These scanners can also operate as film cameras and thus serve as both input and output devices. Monitor stations have also been attached to the Input/Output System. These each consist of a CRT display, a typewriter, and a magnetic tape unit; and are provided to assist human control of the analysis.

The duty of the Pattern Articulation Unit (PAU) is to perform local preprocessing on the input from the scanners, such as track thinning, gap filling, line element recognition, etc. The logical design of this all-digital processor has been optimized for the idealization of the input image to a line drawing. Nodes representing end points, points of inflection, points of intersection, etc. are labelled in parallel by appropriate programming under overall control of the Taxicrinic Processor. The abstract graph describing the interconnection of labelled nodes is then extracted as a list structure, which comprises the normal output of the Pattern Articulation Unit.

This output is then operated on by a Taxicrinic Processor (TP) which assembles such graphs into coherent list structures subject to a recognition grammar and then syntactically categorizes them to complete the visual recognition process. The Taxicrinic Processors are primarily responsible for the execution of user programs, that is, to oversee the operations of the Pattern Articulation Unit, the Arithmetic Unit and to initiate input/output operations in the Input/Output Processors by making requests to the Interrupt Unit.

The Arithmetic Unit (AU) is used exclusively for performing arithmetic operations for the TP. Although there are a few simple arithmetic operations which can be done in a TP (e.g., integer addition) the more complicated operations are done in the AU. The AU has been optimized for double-word floating point arithmetic.

The Interrupt Unit (IU) handles all the interrupt requests from the TP and IOP. When an interrupt is requested it notifies the proper processors which then take appropriate action.

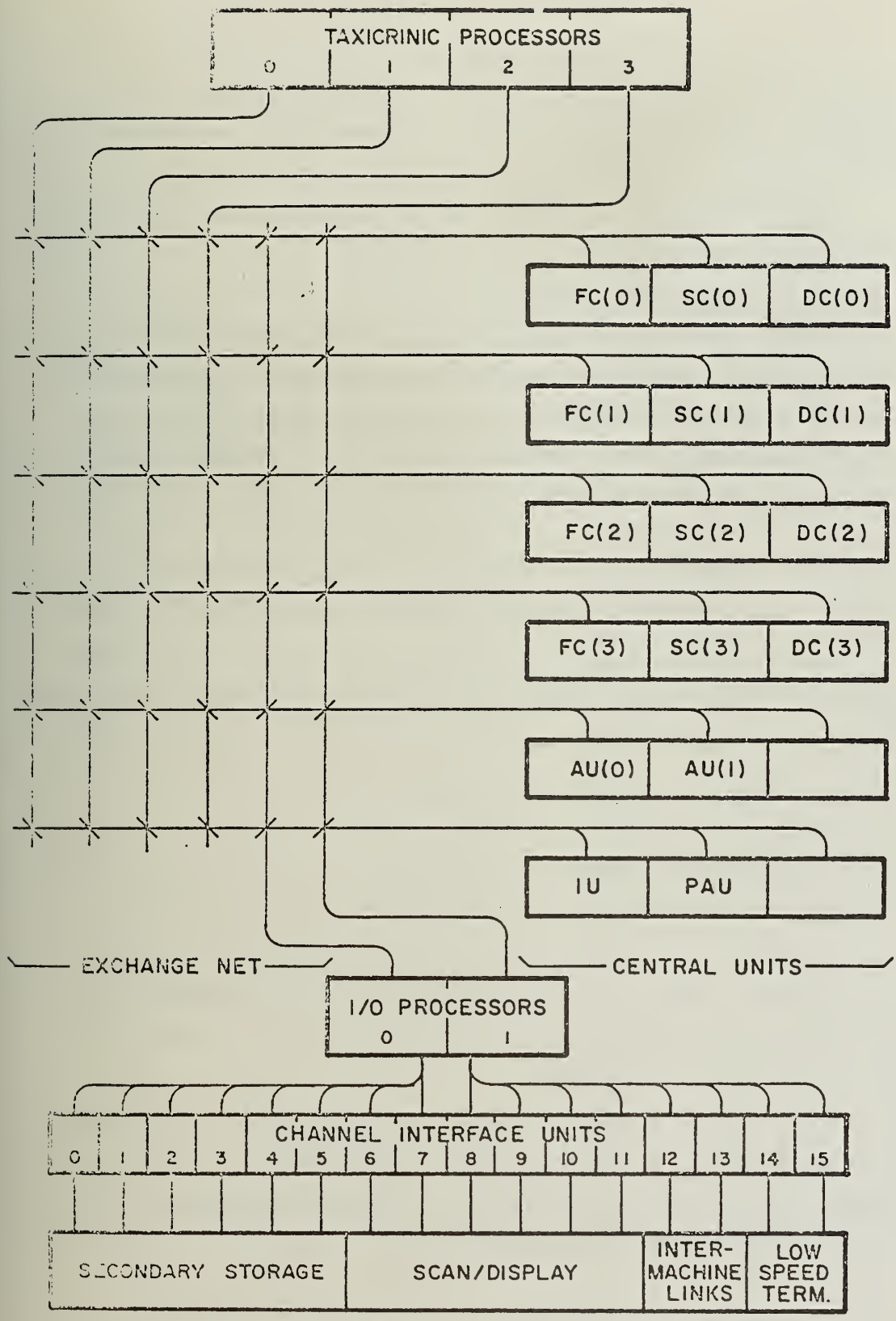


FIGURE 1 – SCHEMATIC OF ILLIAC III COMPUTER

All of the Illiac III processors and units communicate with each other through the Exchange Net (EN) as shown in Figure 1. The Exchange Net is responsible for all the necessary queueing and priority checking.

In the Illiac III system all major modules are designated as either "processors" or "units" according to their position in the Exchange Net. In Figure 1, the processors are shown at the top and bottom and the units are shown on the right. The mirror convention indicates allowable communication paths. The effect of this division is that processors may communicate directly with units and vice versa but may not communicate directly with each other. A processor which needs to communicate with another processor must get help from a unit (normally the Interrupt Unit). In like manner a unit (e.g. the PAU) which wishes to communicate with another unit (say a storage unit) must transfer the information through a processor (the TP in this case).

The Illiac III Computer System can, in summary, be described in terms of 14 constituent subsystems of processors, units and peripheral device groups as follows (see Figure 1):

i) Central System (6 subsystems)
ii) I/O System (3 subsystems)
iii) Peripheral System (4 subsystems)
iv) Power Distribution System (global to all subsystems).

Only two, the Fast Core Store and the Low Speed Terminal Network, are of commercial design. The other constituent subsystems have been designed almost entirely as part of the graduate research program of the Department of Computer Science.

This massive design/development program was not undertaken lightly. Rather, central to the contract program has been the study of the computer architecture appropriate for high speed image processing. These peripheral devices, units and processors must be able to talk to

one another--and with minimum data format conversion: the Scanners feed the Pattern Articulation Unit, whose output in turn is interpreted by a Taxicrinic Unit. This global integration of the design has been costly but integral to the integrity of the system design.

Each following section summarizes briefly each constituent subsystem: Taxicrinic Processor, Arithmetic Unit, etc. In general a function description is followed by a discussion of special design features (if any) and available documentation. Documentation listed is that presently current with the exception that occasionally an earlier report is listed if required for conceptual continuity.

List processing has also been designed into the processor by including instructions for the control of Available Space, and for the insertion/deletion of a list cell.

In addition the TP must interpret Pattern Articulation Unit instructions, much as a more conventional CPU must communicate with an Arithmetic Unit. Where possible operand transmittal has been based upon buffering in the Operand Stack of the Taxicrinic Processor, an array of 32 bytes of fast storage with automatic refill/unload facilities coupling to that portion of the stack in core.

A maximum of four TP's are allowed by the system. All are exactly the same: each is equipped with its own private fast registers and operates completely independently of the other TP's.

1.1.3 Documentation

Report

Report No. 417

Nordmann, Bernard J., "Illiac III Computer System Manual - Taxicrinic Processor Volume 1", Revised from Report No. 341. November 24, 1970.

Nordmann, Bernard J., "Illiac III Computer System Manual: Taxicrinic Processor Volume II" (To be published).

1.1.4 General Organization of the Taxicrinic Processor

1.1.4.1 The Taxicrinic Processor

Figure 1.1.4.1 shows the main registers and subprocessors of the TP, exclusive of the main control, together with the various data paths between them. The main "active" registers are the Instruction Register (IR), the Logic Register (LR), the Distribution Register (DR), the Arithmetic Register (AR), and the Spare Buffer Register (SBR). These are all 36 bits long (4 bytes of 8 bits and 1 flag each) and are used to hold data temporarily as various instructions are being processed. None of these are directly accessible to the programmer.

The main "storage" registers are the Base Registers (8 registers of 27 bits or 3 bytes), the Associative Registers (7 registers of 16 bits), the Pointer Registers (15 registers of 36 bits or 4 bytes), the PR Segment Name Registers (16 registers of 16 bits), the Operand Stack (one continuous register of 32 bytes) and the Instruction Buffer Register (one double word of 8 bytes). These registers contain data which may be used by the programmer during the execution of his program. They are all accessible by programming except for the Associative Registers and the Instruction Buffer Register, although the Base Registers may only be changed indirectly by the Operating System.

There are other registers in the TP which are not accessible to the programmer and which are used during special operations. These will be fully explained in their respective sections in this manual.

Finally in connection with Figure 1.1.4.1 it should be noted that the output from the Permuter, the Distribution Bus (DB), is not a register. It consists of 36 lines which serve as inputs to nearly all of the registers and control sections in the TP. The DB is the main avenue of information transfer in the TP.

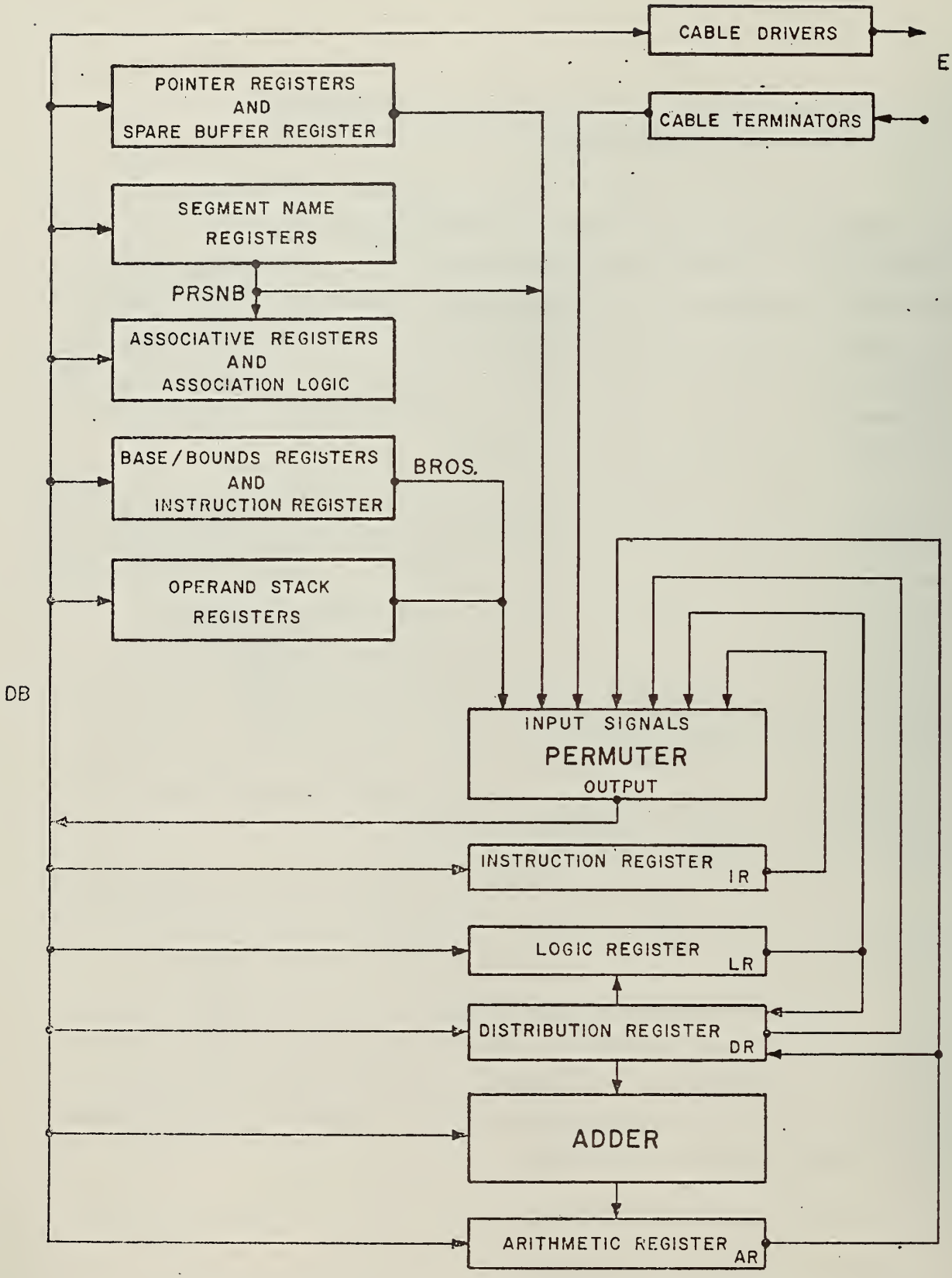


Figure 1.1.4.1 - Major T.P. Registers and Data Paths

1.1.4.2 The TP Subprocessors

The general structure of the TP is divided up into various subprocessors which are in turn controlled by the TP main control sequencing. These subprocessors communicate with the main control by means of "control lines" which give commands to the subprocessors and "output lines" which give the main control information about the status of the subprocess. The subprocessors are in general fairly independent and need only receive a small number of control line inputs to perform rather complex operations.

The main subprocessor groups are the Permuter, the Operand Stack and Operand Stack Control, the Pointer Registers and Pointer Register Control, the Base Registers and Base Register Control, the Instruction Buffer Register, the 32 Bit Adder, the Boolean/Shift Logic, and the Algebraic/Logical Compare Logic.

The Permuter is a system of gates and drivers which is central to the transfer of information in the Taxicrinic Processor. As can be seen in Figure 1.1.4.1 it is capable of transferring information to and from all the subprocessors of the TP and essentially all of the fast storage registers.

The name "permuter" is derived from the fact that an input cell may be permuted to make its boundaries coincide with its destination boundaries. In addition to permuting the input cell, the permuter has the capability to inhibit all of the output bytes in any combination, or any combination of bits in the rightmost output byte (byte 3). This facility enables the permuter to generate constants to put on the distribution line and to mask in bits to the low order positions of, say the Pointer Registers, as they are gated through the permuter.

The actual Permuter logic cards are also used to construct the storage space for the LR, IR, AR and DR. The Permuter is described in Section 1.1.4.6.

The Subprocessors for the Operand Stack, the Pointer Register and the Base Register all require a more detailed description than could

possibly be given at this time; their complete description will be deferred until Section 2.

The Instruction Buffer Register (IBR) is an eight byte (double word) register used to store that portion of the instruction code which has yet to be processed. It may also be used to store successive instructions which will be executed after the present one.

The 32 bit Adder in the Taxicrinic Processor is used to perform binary addition within the TP. It is also used to generate the outputs for the Boolean operations, EQV and XOR. The adder uses 2's complement number representation and employs two levels of carry lookahead to hasten carry propagation. The adder is broken down into eight four-bit sum groups with full carry lookahead within the groups. The second level of lookahead occurs between groups, with lookahead between groups 1, 2 and 3 and between groups 4 through 8. The final carry between these two second level groups is a "ripple" carry.

The Boolean logic performs the Boolean functions "AND", "OR", "XOR" and "EQV". The "XOR" and "EQV" functions are generated using part of the 32 bit adder.

The Shift logic employs the permuter to make shifts in multiples of 8 bits. After this has been done the shift control shifts one bit at a time until the proper number of additional bits have been shifted.

The Algebraic/Logical Compare Logic is used to accomplish algebraic and logical compares (greater, equal and less), the AR equal to zero test, and flag match compares. The instructions realized are CPRA, CPRL, TEST(M), SCAN(M), and TA. Operands are compared by subtracting one from another; then signs are compared and appropriate indicators are set. The logic discussed here can only be used with long or short fixed point numbers; decimal and floating point numbers are routed to the floating point arithmetic unit.

1.1.4.3 Base Registers

The Base Registers (BR's) are used as a part of the memory accessing scheme to contain information about the segments of the current process. Up to 2^{14} segments are allowed the programmer. But as there are only 7 usable base registers, only the data on the 7 most recently used segments is held within the TP. The remaining segment base information is retained in memory in the Segment Table of the current process. BR's 1 through 7 are used to address recent data. BR#0 is used to indicate the Segment Table.

To identify which segments have their bases in the TP, each base register (other than BR#0) has an associative register assigned to it which contains the name of the segment currently in that base register. When a PR is selected as an address of a cell to be accessed from memory, its segment name register is associatively compared with the associative registers for each base register. If a match is found, the matching base register will be used. If there is no match, the required base data will be accessed from the Segment Table.

In order to completely determine a file, the base register must contain:

1) a base address which indicates where the file storage begins. (In Partitioned Mode the base address is the address of the page map - see Section 1.1.5).
2) a bounds, which indicates how many pages have been assigned,
3) a code bit to indicate the Data Access Mode (Contiguous or Partitioned), and
4) two other code bits to determine the level of Access Privilege (read-write, read only, trap, or no access).

The base address uses a 16 bit page address which represents the upper 16 bits of a normal 24 bit address. The rightmost 8 bits of the corresponding absolute address are always taken to be zeros. The bounds is an 8 bit quantity allotting the user a specified number (up to 256) of pages of memory for this file. These two quantities are placed in a 3 byte BR with the bounds in the leftmost byte, the base address in the rightmost 2 bytes, as shown in Figure 1.1.4.3. The Data Access Mode bit is in the flag position of the leftmost byte and the Accessing Privilege bits are in the flag positions of the 2 rightmost bits. How these various quantities are used is explained in Section 1.1.5 on Memory Accessing.

Since the base information controls where and how a user may access the memory, users must be prevented from changing the contents of any BR. For this reason operations which change the value of any portion of a Segment Table are considered "privileged" and may only be used by the supervisor routine.

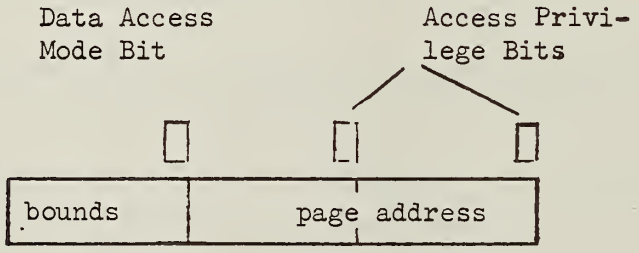


Figure 1.1.4.3 Base Register Format

1.1.4.4 The Pointer Registers

There are 15 Pointer Registers (PR's) in the Taxicrinic Processor. These registers are in some ways similar to the index registers of more conventional machines, although they are much more powerful. As with conventional index registers, the PR's may be incremented, modified and tested. Thus they can be used in many of the ways that an index register might be used, except that these operations usually appear quite different in the TP because of the unusual machine organization.

Primarily, however, the PR's are a means of addressing storage. All storage accessing is performed "through" the PR's, by indicating the name of the PR which contains the address desired instead of requesting the actual address (the contents of the PR can be modified as noted above). The indicated PR will contain a segment name and an address in that segment relative to the segment base address. By loading the PR's beforehand, the programmer can refer to any address in memory by using only a 4-bit tag to indicate the name of the PR to be chosen. (For a more detailed description of memory accessing, see Section 1.1.5).

In order to be able to choose a particular PR, each PR has associated with it a 4-bit Name Register (NR). Each NR holds the current "name" assigned to its associated PR. These names are unique but not permanent, i.e., the name of the PR (which is a number for 0 to 14) can be changed, but only one PR has any given name at any given time. The process whereby the PR names are changed is called "name permutation" and is described in detail in Section 2.2.10.6.

Since the PR's are the only means by which memory may be accessed, the address of a variable must be assigned to its particular PR prior to its first call. Provisions are provided in the instruction set to initialize all designated PR's either one-at-a-time or by multiple assignment (using an imprimitive instruction, see Section 2.2.10.)

There are three formats for the pointer registers. The "normal" format is used in connection with the pointer stacks as shown in Figure 1.1.4.4.1 of Section 1.1.4.4.1.

The "list processing format" is used with the list processing instructions and does not utilize pointer stacking. Instead the pointer is divided into two half-word link fields which are identical to the contents in the cell currently being "looked at" by the pointer register.

The third format for pointer registers is the "available space" format. This format is used whenever a PR is utilized to control the assignment of list processing cells. In particular PR14, which controls the storage of the pointer registers in the pointer stacks, is in this format.

1.1.4.4.1 Pointer Stack Format

Reflecting the nested nature of operand calls, pointer stacks are provided into which the current value of a PR can be pushed for safekeeping. When this value is again required, the stack can be popped and the value reloaded into the PR. The pointer stacks are maintained in main storage under the control of PR14, which is in the "available space" format (see Section 1.1.4.4.3).

The pointer stacks themselves are ordinary unidirectional linked lists. The pointer stack cells take up two words of storage, but only the first 6 bytes are currently being used. As shown in Figure 1.1.4.4.1, the first two bytes are used to hold the 16 bit address of the previous entry in the stack, while the next two bytes contain the relative address within the segment. The final 2 bytes contain the segment name.

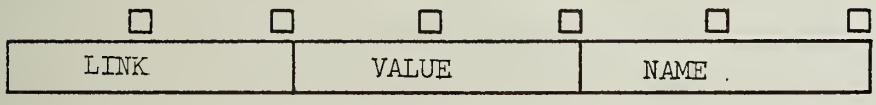


Figure 1.1.4.4.1 Pointer Stack Cell

The physical Pointer Registers in the TP actually consist of 2 registers, a 4 byte register containing the link and value fields and a 2 byte register containing the name field. This configuration of the pointer register is referred to as the "normal" or "pointer stack" format. It should be noted that all flag bits are always pushed into the pointer stacks along with the link, value and name fields.

1.1.4.4.2 List Processing Format

The List Processing format for the pointer registers differs from the normal PR format. As shown in Figure 1.1.4.4.2 it consists of a left and right link instead of a link and value field. The name field remains the same and indicates the name of the segment being used as the base for the two links. There are no stacks connected with PR's in this format.

Figure 1.1.4.4.2 List Processing Format

In the list processing format, a PR can be considered to be a "bug" which moves along a list structure in core. At any given time the PR contains an image of the first four bytes of the cell currently being "looked at". These first four bytes will usually be interpreted as two separate pointers. It should be noted that the PR does not contain a pointer to the cell being looked at. It only contains the contents of the first 4 bytes of that cell. In the conventional method of using the Illiac III list processing instructions (specifically SL and SR), the address of the current cell will be available at the top of the OS.

In use the PR can be used to sequence through a list structure. At any given point, cells may be added or deleted to the "right" or "left" of the current cell. Links in the structure may also be changed by using the ASSIGN statement to load the contents of some PR into a cell in the structure. Instructions such as POP and PUSH can also be used to load and unload links from the OS.

1.1.4.4.3 Available Space Format

The Available Space format is used whenever a PR is utilized to control the assignment of list processing cells. PR#14, which controls the storage of pointer registers in the pointer stacks, is always in this format. As shown in Figure 1.1.4.4 3/1 the pointer register in the available space format consists of a link field and a count field. The name field retains the same interpretation as for the normal and list processing formats, and indicates the name of the segment being used as the base for the link and count relative addresses.

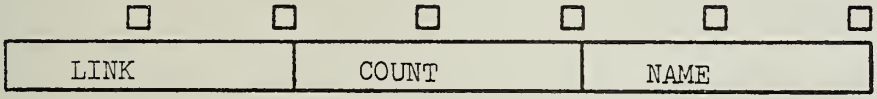


Figure 1.1.4.4.3/1 Available Space Format

The basic idea behind this format type is to set aside an available space file in memory to be used in the construction and destruction of list structures. Automatic allocation features of this file — when using the "available space" format for the associated PR — allow the programmer to obtain empty cells and to discard unneeded cells.

An available space file may be divided up into two parts: a section of list storage occupying the low address end of the file and a section of consecutive storage located in the high address end of the file. The list storage part consists of a sequence of cells; here the first 2 bytes of each cell contain the link to the next cell while the rest of the cell may eventually contain data. The size of these two areas changes dynamically as the file is used. At the very beginning of use the file is completely made up of consecutive storage, but as more storage is needed for the linked lists,

this area is absorbed into the list storage area. Figure 1.1.4.4.3/2 gives an example of the appearance of the file after it has been in use awhile.

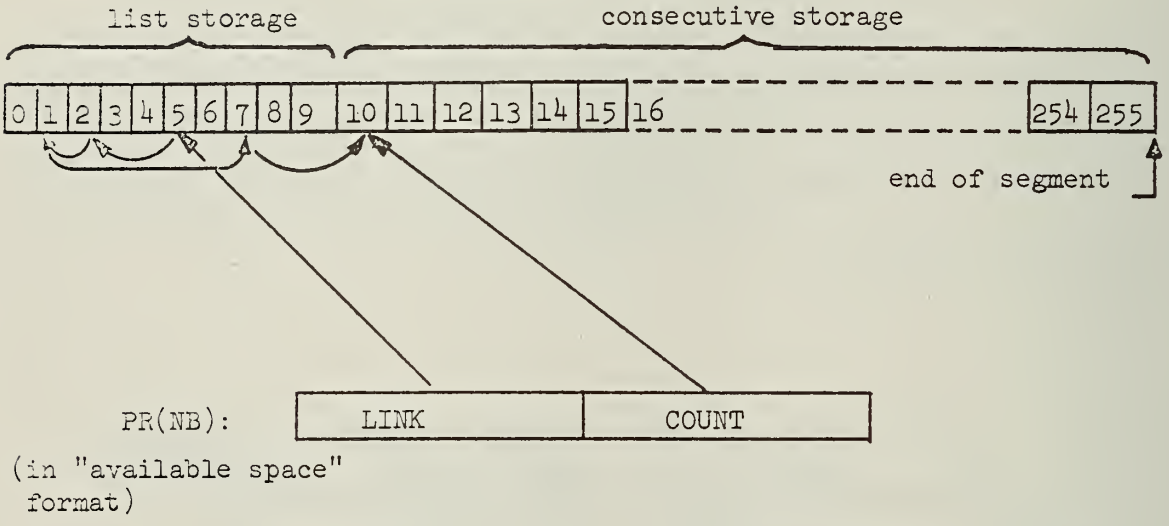


Figure 1.1.4.4.3/2 Available Space File

Referring to Figure 1.1.4.4.3/2 whenever a list cell presently assigned to a user is no longer needed by the user (i.e. it is "popped" off the user's list), it is placed on a special list (in the list storage area) called the "free list". This list contains all the cells in the list storage area which have been "freed" to be used as storage for other lists. The Pointer Register (in "available space" format) contains a link to the top cell on this list so that all access to the free list is made through the available space PR. In the example of Figure 1.1.4.4.3/2 the free list link would equal the address of cell 5. Cells 5, 2, 1 and 7 are on the "free list".

Whenever data needs to be stored on one of the user lists in the list storage area, the available space PR is modified so that it is

linked to the second available free list cell, and the top cell on the free list can then be used to store the new data and its link. For example in Figure 1.1.4.4.3/2 if a cell were needed by one of the user lists, the free list link of the available space PR would be loaded with the link portion of cell 5, i.e. the address of cell 2 in this case. Then cell 5 could be used by the list which needed a new cell.

Only cells of one size may be placed in any given available space file. The size of the cells in a given file is determined by a number initially placed in the leftmost halfword of the zeroth cell in the file. (When using PR14, the cell size is known to be 4, so this latter procedure is ignored.) In general any even integer ≤ 256 may be used for the cell size (for other than PR14). This zeroth cell is never distributed since its address would conflict with the symbol for the end of a list (i.e. link field of zero). Accordingly an available space pointer is normally initiated with LINK = COUNT = cell size > 0.

But what happens if the free list is empty? To solve this problem the last cell in the free list (#7 in the illustration) has its link pointing to the lowest addressed cell in consecutive storage. The second two bytes of the available space PR (called the COUNT) also contains this address, so that whenever the LINK equals the COUNT, the free list has been exhausted. When this happens and a request for another cell is received, the first cell of consecutive storage is used and both the count and link are incremented by the number of bytes in the cell. (However, if the assigned cell crosses or exceeds the segment boundary, an Available Space Empty interrupt is generated. In this latter case the count and link are not incremented, and therefore reflect the arrangement of the available space at the time of the illicit memory access.)

As an example, in Figure 1.1.4.4.3/2 if the free list became exhausted, the LINK and COUNT would both be equal to the address of cell 10. If a request for a cell were then made, cell 10 would be given out and both LINK and COUNT would be incremented by the value stored in the first halfword of the zeroth cell. LINK and COUNT would then point to the address of cell 11.

The count receives its name from the fact that it is the number of cells (times the cell size) in the list storage area. When the count equals the cell size, all storage is consecutive. (The zeroth cell is dedicated to the cell size.) When the high order byte of the count equals the address bounds of the segment all storage in the segment is of the list type.

One precaution should be noted. Since a PR in available space mode does not have a pointer stack of its own, it cannot be pushed into one. As a result a programmer must be extremely careful to avoid pushing an available space format PR or drastic confusion may result. In general there is nothing in the hardware to prevent this.

1.1.4.5 Operand Stack

The Operand Stack (OS) is a specially designated file in the main store defined by pointer register #13. It gains its distinction by virtue of having its topmost bytes (up to a maximum of 32) located in fast access storage registers internal to the Taxicrinic Processor. For this reason the Operand Stack can be used as a scratch pad for evaluating arithmetic and boolean expressions with a minimum access to core storage. Many TP instructions obtain operands from or operate directly on the OS. Some instructions, such as LD or PØP, merely transfer data cells into or out of the OS. Others, like ADD, pop out two cells, operate upon these operands and then return a single result.

Programming conventions established for the OS make it appear that the stack is at all times entirely in core. It can be visualized as a continuous string of bytes. The Operand Stack Pointer (Pointer Register #13) can be considered to contain the 16-bit relative address α of the first (leftmost) unfilled byte in the stack, relative to its associated base.

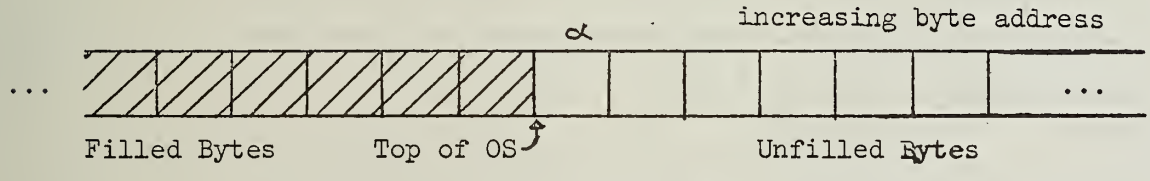


Figure 1.1.4.5/1 Operand Stack Schematic

An operand can be pushed into the stack by assigning it to the field starting at location α, and then incrementing the OS pointer by the field (or cell) size. In like manner, an operand may be popped from the stack by decrementing the OS pointer by the field length (i.e. 1, 2, 4 or 8 bytes). Operand boundaries are by no means sacred: for

example, fields pushed into the stack as bytes may be popped out a word at a time.

To add or delete from the top of the stack we must:

1) specify the field length of the operand by giving a "cell size" parameter as part of the instruction. The cell size parameter consists of two bits and identifies the operand as a byte (00), halfword (01), word (10) or double word (11).

2) distinguish whether the operand is to be pushed into the stack (at the address = the top of the stack) or popped from the stack (at the address = the top of the stack minus the cell size).

Although the top bytes of the Operand Stack are located in fast registers in the TP, any operand in the stack may be accessed by using PR #13 as a file address and suitably modifying it to get the desired byte within the stack. This request necessitates emptying the contents of the TP fast registers into core first. The complete stack will then indeed be in core, as the programmer expects. Of course this process involves time; however it is not anticipated that this method of accessing the Operand Stack will be used often, so this slow-down should not be significant.

The programmer will most often visualize the stack as a simple list of items (..., LEFT 3, LEFT 2, LEFT 1), each item a fixed-length operand (see Figure 1.1.4.5/2 below). (In addition if all fields are the same length, the stack is called uniform.) It must be realized, however, that LEFT 3 (say), though an appropriate symbolic address in an assembly program, is not a machine-recognized address: the address of

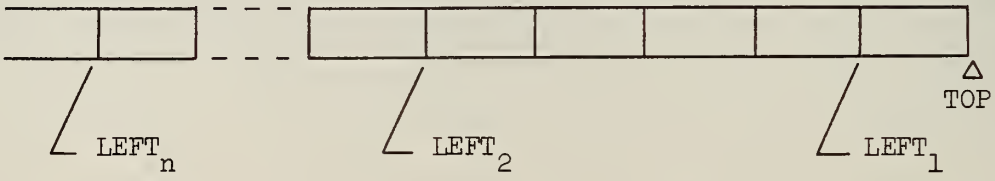


Figure 1.1.4.5/2 Convention for Naming Operands within a Stack

the field LEFT 3 depends not only on the field length of LEFT 3, but also on the field length of LEFT 1 and LEFT 2. In any case, binary and ternary operators defined on the contents of the stack may merge stack operands.

1.1.4.6 Permuter

The Permuter is a system of gates and drivers which are central to the transfer of information in the Taxicrinic Processor. As can be seen in Figure 1.1.4.1, it is capable of transferring information to and from all the subprocessors of the TP and almost all fast storage registers.

The name "permuter" derives from the fact that an input cell may be permuted to make its boundaries coincide with its destination boundaries. A permutation begins with a circular left shift on a byte basis. The position of the four bytes is changed, but the cyclic ordering of the bytes is preserved. For example, the cell:

A	B	C	D

appears as

B	C	D	A

when permuted left one byte and as

D	A	B	C

when permuted left three bytes. Cells smaller than 4 bytes (that is, bytes and half-words) are assumed to have garbage in the non-data positions, while double words must be permuted one word at a time.

In addition to permuting the input cell, the permuter has the capability to inhibit all of the output bytes in any combination, or in the rightmost output byte (byte 3) to inhibit any combination of bits. This

facility enables the permuter, among other things, to generate constants to put on the distribution bus (±32, ±8, ±4, and ±2 are provided for) and to mask in bits to the low order positions of (for example) the pointer registers as they are gated through the permuter.

1.1.4.7 Instruction Buffer Register

The Instruction Buffer Register (IBR) is an eight byte (double word) register used to store the instructions as they are obtained from core. The Instruction Register (IR) is used to hold that part of the instruction which is currently being processed and is loaded from the IBR. In the case of Primitive Instructions the IR usually contains the complete instruction.

1.1.4.8 The 32 Bit Adder

The 32 bit adder in the Taxicrinic Processor is used to perform binary addition within the TP. It is also used to generate the outputs for the boolean operations, EQV and XOR. The adder uses 2's complement number representation and employs two levels of carry lookahead to hasten carry propagation.*

The inputs to the adder are obtained from the Distribution Register (DR) and the Permuter Distribution Bus (DB). These two inputs can be gated to the adder only in "true" form. To calculate the difference of two numbers, the subtrahend must be gated from the 9 bit/byte storage (Operand Stack, Pointer Registers, etc.) in 1's complement form (see TP Manual for details). In this case a carry is injected at the low-order end of the adder to obtain the true 2's complement difference.

The adder itself is broken down into eight four-bit sum groups with full carry lookahead within the groups. The second level of lookahead occurs between groups, with lookahead between the leftmost groups 1, 2 and 3 between the rightmost groups 4 through 8. The final carry between these two, second level groups is a "ripple" carry.

---

*For a more complete discussion of carry lookahead adders see Wiegel, Roger E., "Methods of Binary Addition", DCS Report No. 195, February 1966.

1.1.4.9 Boolean/Shift Logic

The Boolean logic performs the Boolean functions "AND", "OR", "XOR" and "EQV". The "XOR" and "EQV" functions are generated using part of the 32 bit adder.

The shift logic employs the permuter to make shifts in multiples of 8 bits. The shift control first makes the highest multiple-of-8 bit shift that it can without exceeding the desired shift. This is done using the permuter and inhibiting the necessary bytes. After this has been done the shift control shifts one bit at a time until the proper number of additional bits have been shifted. Since the flags are not touched in the bit shifting, the shift control will shift flags in multiples of 8 only (i.e., when the permuter is used).

1.1.4.10 Algebraic/Logical Compare Logic

The algebraic/logical compare logic in the TP is used to make comparisons between byte, halfword or word size cells. If it is necessary to compare two floating point or BCD numbers, this is done using the Arithmetic Unit.

Generally speaking the comparisons are made by subtracting and then comparing the result with zero. Therefore the hardware is set up to test the AR for zero. Various groups of bits are tested depending on the cell size and the type of comparison. The sign is also checked to set the "greater" and "less" flip-flops.

The compare is usually done in two cycles. During the first the "greater" or "less" flip-flops are set according to the signs and the EQ is set to 1 if the AR is zero. In the second cycle, both the "greater" and "less" flip-flops are reset to zero if the EQ flip-flop was previously set to one.

1.1.5 Memory Access

The main task of the Memory Access Sequence is to obtain data. The data may be of the immediate type, in which case no access to core is necessary, or it may be non-immediate, in which case one or more accesses to core will be needed. If access to core is required, the given virtual address is automatically checked against the contents of the bounds field of the base register. The alignment of the cell with boundaries appropriate for the given cell size is also checked. The address in core is then constructed relative to the base address. If the base information for the segment being requested is not in one of the seven "active" base registers, the Memory Access Sequence will automatically access the Segment Table, load the proper base information into the least-used base register and continue with the access.

There are two distinct memory organizations, both of which are hardware implemented. In the Contiguous Data Access Mode all memory accesses are made through the pointer registers directly to the user's file by means of the base information. The Partitioned Data Access Mode makes use of a page map (in memory) which is accessed first to identify the page in memory which is to be accessed. This procedure allows for a more flexible storage organization at the cost of a slower access time. Both the Segment Table and the segments themselves may be partitioned.

1.1.5.1 Contiguous/Partitioned Storage Organization

There are two types of memory organization in the Illiac III system: contiguous and partitioned. As far as the user is concerned virtual memory is conceived as contiguous, independent of the selected mode of memory organization.

In the Contiguous mode each segment is referenced by its base information contained in a base register. This information consists of the page address of the first page in the segment, the length of the segment and the access privilege of the segment. Each page in the segment must be contiguous with the rest of the segment and all of the pages have the same access privilege.

In the Partitioned mode the various pages of the segment do not have to be contiguous. Pages can be stored in distinct areas of memory (possibly units having different access times), with each page having its individual access privilege.

The Partitioned mode provides several advantages in storage allocation:

1) It is much easier to find storage for large files since the files do not have to be contiguous.

2) There is never a need for a complex "garbage collection" ob of blocks of storage which have been discarded by various processes, since there is no need to try to combine them into the largest contiguous blocks possible.

3) If a file, at some time subsequent to its initial storage allocation, requests more space, it can be conveniently taken from any unassigned pages in storage.

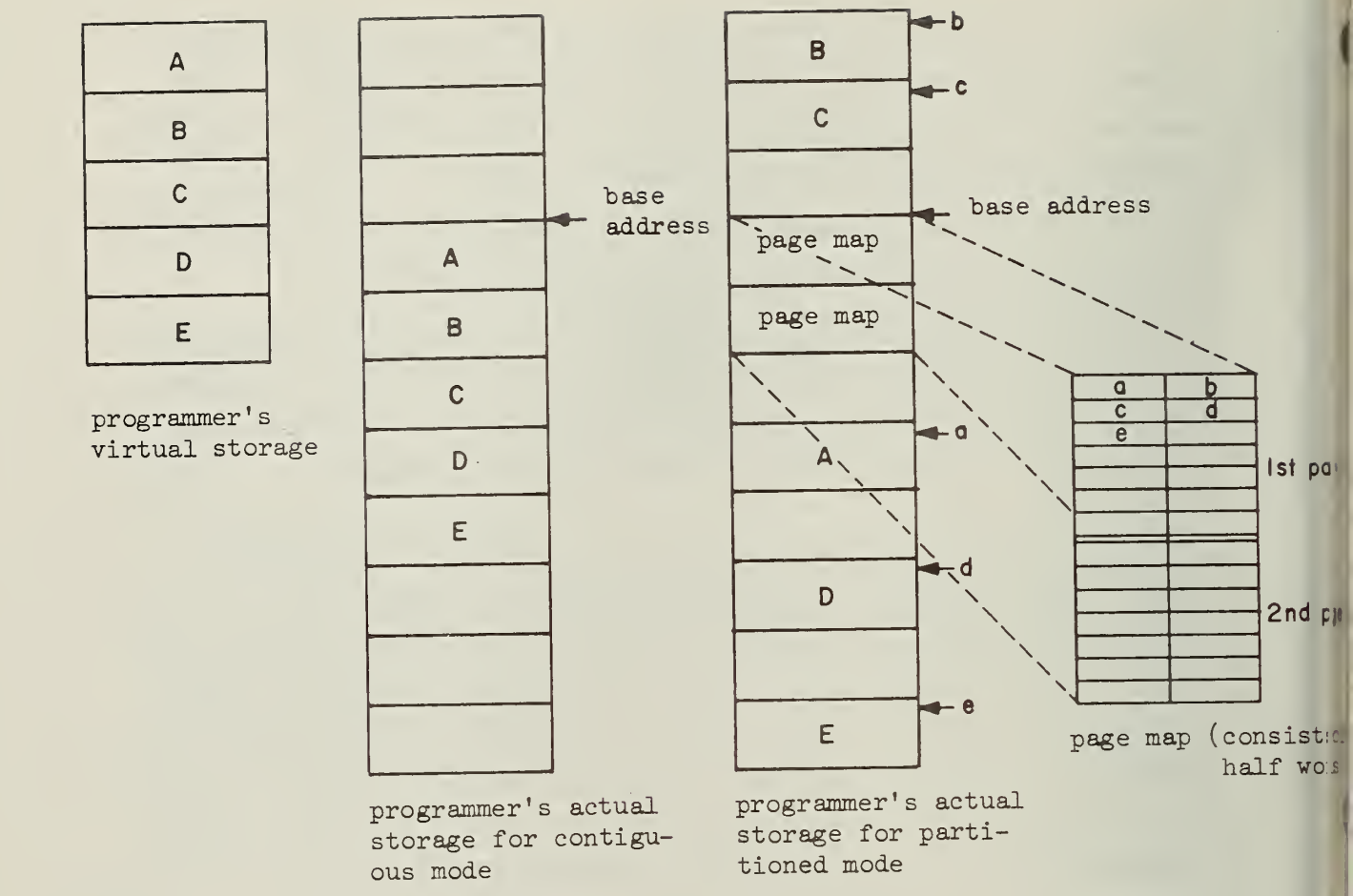
The basic idea behind the partitioned mode is to take more efficient advantage of storage by discarding the need for contiguous storage. To allow a file to maintain storage pages in an arbitrary number of places in memory (up to 256 pages) however, a strategy is

required to keep track of these page locations. The memory access strategy should demand a minimum of effort on the programmer's part, and as little extra hardware and time overhead as possible. To these ends the following decisions were made:

1) The implementation of the Partitioned Mode of memory accessing will be completely in hardware.

2) File storage allocation will be done by the Operating System. The programmer need not worry about the mode of memory access unless he specifically desires to request one or the other modes of operation.

To keep track of the pages in the segment, a page map is used. This consists of two consecutive pages in memory which are set up by the Operating System when the user first requests storage allocation. The map contains up to 256 half-words which specify the page addresses in actual storage of the user's virtual memory pages and for each page, its access privilege. These 16 bit half-words are arranged in the order in which the user's pages appear in virtual memory. (See Figure 1.1.5.1). In Partitioned Mode the base information is used to indicate the location of the page map and its access privilege. In common with the Contiguous Mode the bounds field of the base register is used to make sure the address is within the virtual memory assigned to the segment. Each time an access is desired, the control unit checks the first flag bit in the base register for a "1" to see if the Partitioned Mode is to be used. If it is, the page address field of the virtual address, i.e., the page address in the user's virtual memory, is converted to an address of a half-word in the page map. This half-word contains the address in actual memory of that particular page



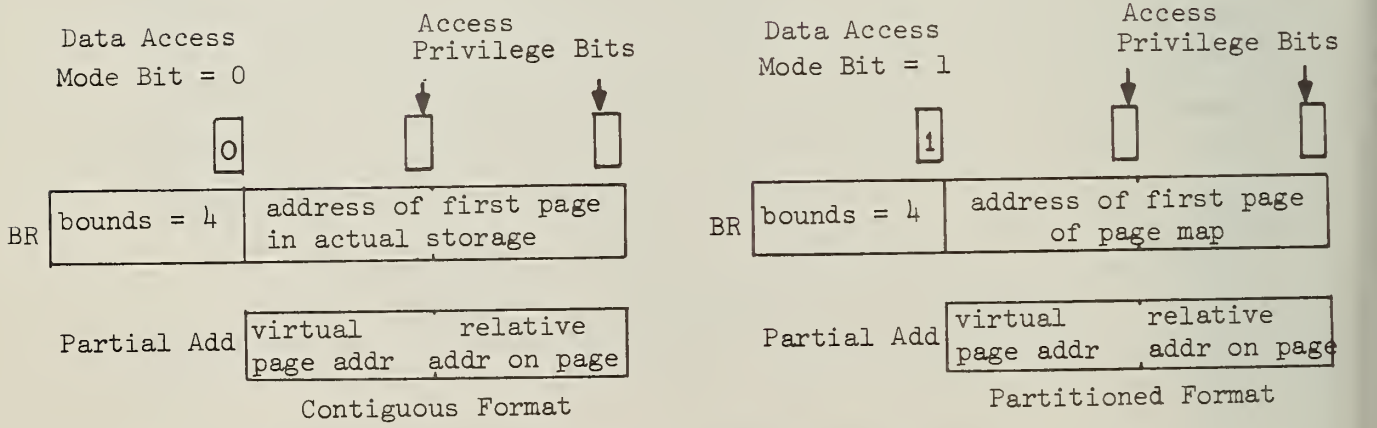


Figure 1.1.5.1 - Pictorial Comparison of Partitioned Mode vs. Contiguous Mode of Data Accessing

Note: Virtual Addresses (those constructed by programmer) are the same. Only BR's are different and these should only be assigned by the Operating System. Programmer will not know the difference.

in the user's virtual storage. After its page location has been accessed, the complete address of the data desired can be obtained by combining the page address with the relative byte address on the page as given in the original virtual address.

1.1.5.2 Address Bounds Check

The purpose of this logic is to examine all outgoing memory addresses to insure that an address falls within the area allotted to the segment and hence, program, in question.

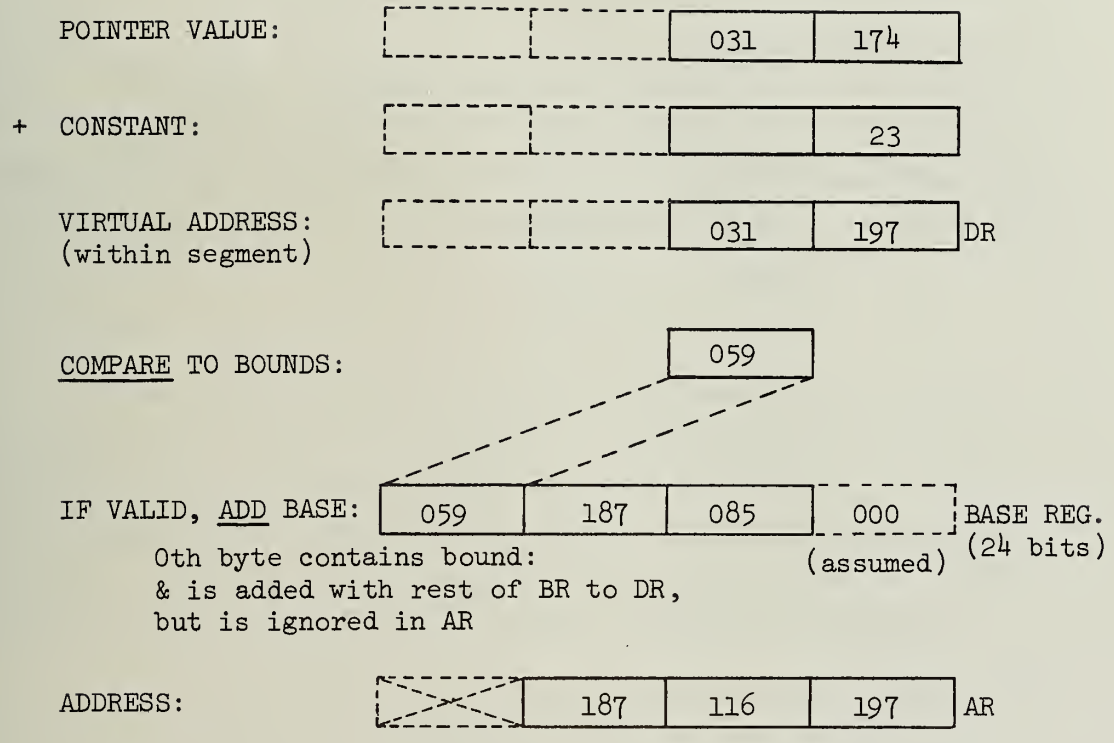
An address is constructed using the base address and either a pointer value field (direct entry) or a preformed constant (modified entry). If the pointer value or constant passes a comparison with the given bounds for that segment, it can be added to the base address (the high order 16 bits of a 24 bit address) to give a final 24 bit address which may then be sent to the core unit. (This is the procedure for a contiguous mode access; partitioned mode will be explained below.) A graphical illustration of the process is given in Figure 1.1.5.2.

The base registers hold a 2 byte (16 bit) base PAGE ADDRESS and an associated 1 byte page count*, or BOUNDS. The page address indicates the beginning of the segment associated with the given base register. The BOUNDS is one less than the number of pages in the segment. This greatly simplifies checking the logic for a relative overflow (see below). The minimum allowed segment length is thus one page, corresponding to a BOUNDS = 0. The maximum is 256 pages (BOUNDS = 255). One page contains 256 bytes, 64 words, or 32 double words.

In the bounds check, the 16-bit virtual address is compared with the given BOUNDS to check for an "absolute" bounds overflow, i.e. the virtual address exceeds the page count.

---

*Actually:(page count) - 1; see text.



Note: This file is allowed 60 pages of storage beginning at page | 187 | 085 |. For this example, the numbers in each byte are the decimal representation of the binary contents, i.e. they run from 0 to 255.

Figure 1.1.5.2 Address Bounds Check

If this condition occurs, the bounds overflow indicator (BØV) is set and an interrupt may be initiated.

For memory access in the Partitioned Mode the bounds check still works satisfactorily. In this case the bounds check method for the Contiguous Mode determines if the user is trying to exceed his allotted virtual storage. Since the user is not allowed to change his page map, there is no danger of the user straying into someone else's storage as long as he remains inside any one of his pages. At the same time, if the user tries to access data which overlaps a virtual page boundary, he will again be stopped because of improper cell alignment.

1.1.5.3 Access Privilege Check

In order to provide file security the access privilege bits are used to identify the types of access permitted to each segment. There may be several levels of control. At each level there are always two code bits which indicate the access mode which will be allowed into the next lower level.

The access privilege code is as follows:

Bit 1	Bit 2	Access Mode
0	0	Trap (no read, no write, interrupt)
0	1	Segment, page map or page not there
1	0	Read only - no write
1	1	Read/write allowed

The access level is provided by the base registers and, by extension, the segment table entries. The rightmost two flags of a base register are used for the access code. If the particular segment referred to by the base information is in continguous mode, the access code refers to the status of the segment itself. Note that in this situation the access code refers to the whole segment.

If the segment is in partitioned mode then the access code refers to the status of the page map which lists the actual base addresses of each page. In this case the two flags in each base address indicate the next level of access control, i.e. the access code for each individual page of the segment.

1.2 Main Store

Rapid access storage for Illiac III consists of three types. The fast core and slow core are conventional core store units. The speed difference between these two is about a factor of 5 overall. This speed advantage is partially compensated, however, by the larger module size of selected slow core: 4 times that of fast core. The dictionary is a 'mostly read' memory which can be used for storing tables, system library, etc. which seldom need changing.

The basic cell format of the main store is the double word as shown in Figure 1.2 below.

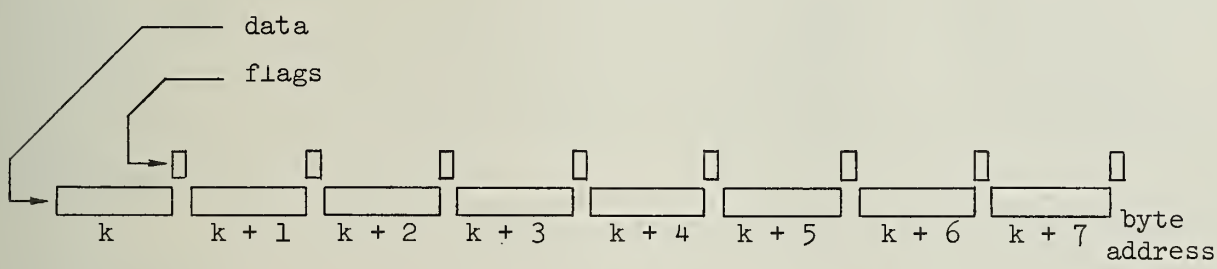


Figure 1.2 Basic Double Word Format of the Main Store
(72 bits/double word; exclusive of parity bits)

Individual byte, halfword, word, and double word fields must begin at a natural cell boundary. A page always designates a field of 256 bytes starting at an address which is an integral multiple of 256 (i.e., the low order 8 bits of the binary address are equal to zero).

Fixed-length fields should be located in the main store at an address which is an integral multiple of the cell size. (In this case, the binary address of a half word, word, or double word will contain one, two, or three low order zeros, respectively). A fixed-length field at an address not an integral multiple of the cell size specified can only be accessed through explicit programming to reassemble the cell. Variable-length fields may start at any location which is a "natural" boundary for the anticipated cell size to be used in interrogating and modifying the field.

1.2.1 Fast Core Store

Two Fast Core Storage Units are presently used in the Illiac III system. Each module has a capacity of 16,384 memory words and a cycle time of 700 μsec.

The standard memory word is 80 bits, consisting of 64 data bits, 8 flag bits and 8 parity bits. This corresponds to an Illiac III double-word cell, augmented by parity bits. Each memory word is divided into eight byte zones which may be read and/or written independently of one another.

Each unit is a self-contained assembly consisting of all registers, timing circuits, power supplies and interface circuitry necessary for operation of the system and compatible with the Illiac III environment. The two units described above were built under subcontract by Fabri-Teck, Inc. and are now operational.

Address range 5,767,168 through 6,291,455 is reserved for fast core storage.

1.2.2 Slow Core Store

Slow core for Illiac III is anticipated to have a cycle time of about 4 μsec. for reading and writing. Compensating however for this slower cycle time is the lower unit cost per byte of this type storage. Accordingly the slow core provides random access mass storage for the system. Initially no slow core storage will be available on the machine.

Address range 6,291,456 through 8,388,607 is reserved for slow core storage. This range represents four times the storage maximally allotted to fast core storage.

1.2.3 Dictionary Store

Dictionary store for Illiac III will be mostly read, occasionally write, storage modules. This store is anticipated to have a slow read time (~10 μsec.), and require approximately one hour to change the contents of the store (i.e., by the insertion of new laser-scanned cards). The imprimitive instructions of Illiac III were introduced primarily to make optimal use of this plug-in dictionary of macros in binary form. Initially no dictionary store will be available on the machine.

Address range 8,388,608 through 16,777,215 is reserved for the dictionary store. This range represents four times the storage maximally allotted to slow core storage, and sixteen times that allotted fast core storage.

1.3 Arithmetic Unit

1.3.1 Function

The two identical Arithmetic Units (AU) perform most of the arithmetic operations in the ILLIAC III system. Integer addition and subtraction, as well as several unary operations form exceptions and are actually executed in the Taxicrinic Processors. (These exceptions are noted in Table 2.3.2/2.)

The prime responsibility of the Arithmetic Units is the high-speed execution of floating point arithmetic operations. The units also provide facilities for integer multiplication and division and conversions from one number-type to another, e.g., floating to long fixed.

As in the case with all other units of the system, communication with the processors is via the Exchange Net. The Arithmetic Units interact primarily with the Taxicrinic Processors (TP), although paths are also available between the AU's and the I/O Processor. The Exchange Net assigns an AU to a requesting processor and also returns the results of an arithmetic operation to the processor which initiated the operation. Figure 1 illustrates the location of the AU's in the overall system.

The following is a general overview of the operation of the Arithmetic Units within the ILLIAC III System. The execution of an arithmetic instruction begins in a Taxicrinic Processor. The operands are assumed to be in the top of the operand stack of the TP. When a TP encounters an arithmetic instruction (an instruction with the mnemonic byte of the form 10XXXXX1*it determines whether an AU is required to execute the operation. If this is the case, the TP places a request with the Exchange Net, which in turn locates and assigns an AU which is not in use. The TP then sends the AU a control byte containing the instruction variant (IV): add, subtract, etc. and the number type (NT): fixed, floating, etc, together with the operands.

---

*The last bit is a flag bit.

Since the data paths through the Exchange Net are one word wide (4 bytes) and since in general, the operands consist of more than one full word, a series of transmissions is required. Upon rapid, initial decoding of the instruction variant and number type, the AU loads the operands as received, a word at a time, into the appropriate registers.

When all operands have been received, the TP-AU path is broken and the AU proceeds with execution of the operation. When the result has been formed, the AU notifies the Exchange Net, which in turn accesses the TP which initiated the AU operation. The result is then returned, a word at a time, to the TP. If an error condition such as OVERFLOW has occurred, a "1" appears on a designated line in the AU to TP control byte. The flags of the erroneous result being returned are set so as to indicate the nature of the error. Upon receiving the result, the TP places it in the top of the operand stack and continues to the next instruction. The AU is released and is available for the next assignment.

The expected execution time for floating point (56 bit mantissa) addition, subtraction, and multiplication is 3-6 μsec. and 8-9 μsec. for division.

1.3.2 Design Features

In keeping with the experimental nature of the Illiac III, the arithmetic units are intended to be a practical testing ground for recent theoretical work in computer arithmetic. Their design uses redundant number systems and a structure in which multiplication and division are executed radix 256. The heart of the unit is the stored-sign subtracter, a recently discovered member of the family of borrow-save subtracters and carry-save adders. A cascade of these subtracters controlled by a multiplier recoder, provides multiplication. The same structure, controlled by a "model division" (a quotient recoder), performs division.

1.3.3 Documentation

Publication

Atkins, Daniel E., "Design of the Arithmetic Units of Illiac III: Use of Redundancy and Higher Radix Methods," IEEE Transactions on Computers, Vol. C-19, No. 8, August 1970, and Report No. 333, May 1969.

Atkins, Daniel E., "Higher-Radix Division Using Estimates of the Divisor and Partial Remainder", IEEE Transactions on Computers, Vol. C-17, No. 10, October 1968.

Report

Report No. 366

Atkins, Daniel E., "Illiac III System Manual: Arithmetic Units," Vol. I, March 1970.

Report No. 418

"Arithmetic Unit of Illiac III: Simulation and Logical Design- Part II", Revised edition, edited by L. N. Goyal, November 1970.

Report No. 397

Atkins, Daniel E., "A Study of Methods for Selection of Quotient Digits During Digital Division," (Ph.D. Thesis), May 1970.

1.4 Pattern Articulation Unit

1.4.1 Function

Image processing can be reduced to the study of those sequences of basic, or root, instructions which permit recognition and description of the input image. These basic instructions can then be implemented in a processor designed specifically for that purpose. Here the duty of the Pattern Articulation Unit (PAU) is to perform local preprocessing on the input from the scanners, such as track thinning, gap filling, line element recognition, and so forth. The logical design of this all-digital processor has been optimized for the idealization of the input image to a line drawing. Nodes representing end points, points of inflection, points of intersection, and so forth, are labeled in parallel by appropriate programming under overall supervision of a control, or taxicrinic, processor. The abstract graph describing the interconnection of labeled nodes is then extracted as a list structure, which comprises the normal output of the processing unit.

1.4.2 Design Features

The PAU employs a two-dimensional Iterative Array (IA) of 1024 (32 x 32) identical processing modules locally connected to execute Boolean functions, threshold logic, and signal path building. It is augmented by its own control unit and by an unconventional core memory, called the Transfer Memory (TM), which in conjunction with the Iterative Array, can operate as an associative memory.

Together the IA and the TM provide a stack of planes for plane parallel picture processing (Figure 1.4.2). In this context the IA can be considered a set of fast scratch-pad registers (of "word-length" = 1024 bits) for the supplementary TM core store (again of 1024-bit planes). In addition the border registers of the IA are explicitly shown in Figure 1.4.2.

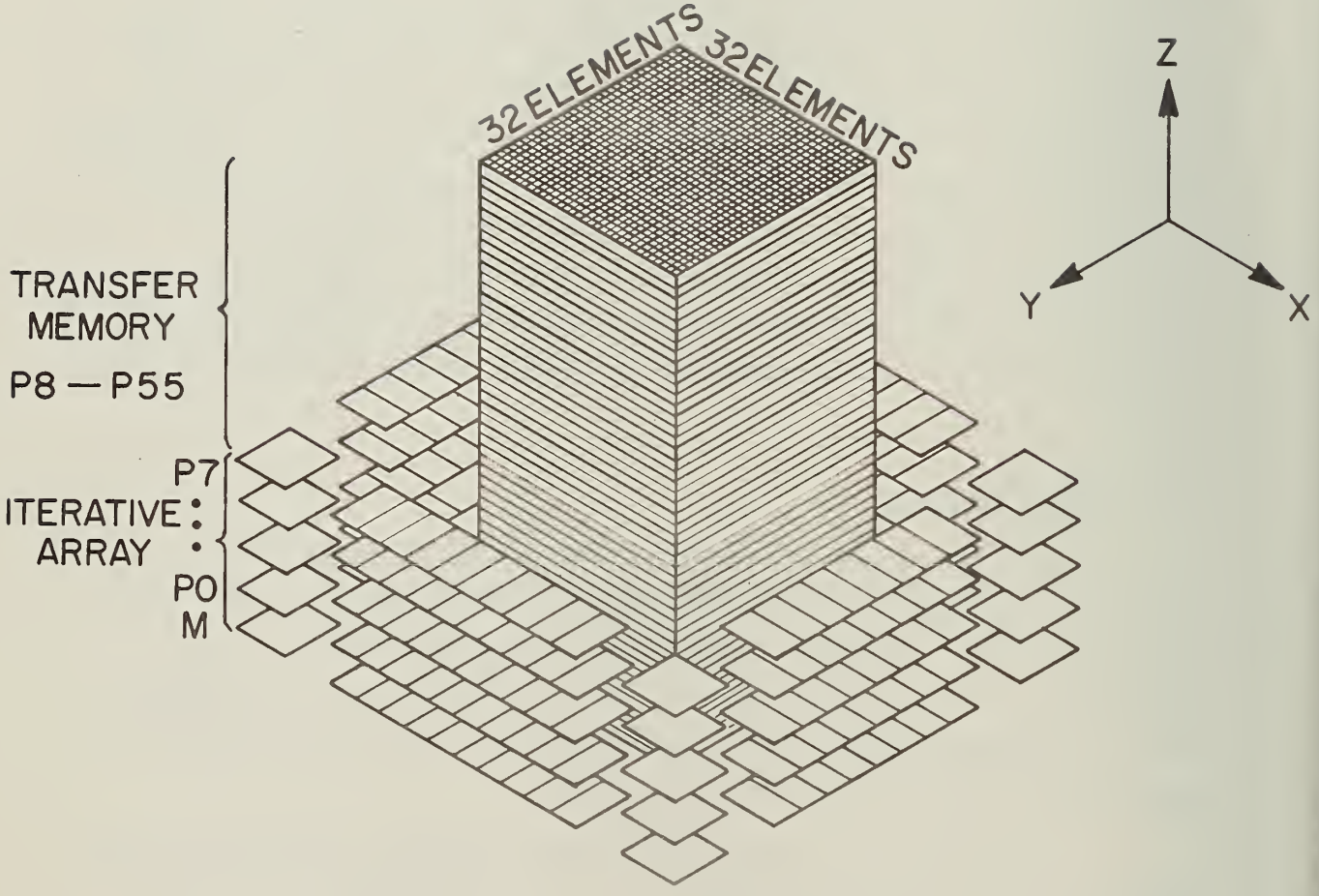


Figure 1.4.2 - Stack of Planes for Plane Parallel Picture Processing: showing Iterative Array/Transfer Memory Interconnection

1.4.3 Documentation

Report

Report No. 314	Borovec, Richard T., "The PAX II Picture Processing System at the University of Illinois: Programming Manual", March 1969.
Report No. 338	Borovec, Richard T., "The Pattern Articulation Unit of Illiac III: Implementation of the Homogeneous Instruction "BOOLE", June 1969.
Report No. 253	Borovec, Richard T., McCormick, B.H., Watson, W.J., "The Pattern Articulation Unit of Illiac III: Homogeneous Boolean Functions in the Iterative Array, January 1968.
Report No. 285	Borovec, Richard T., Koo, Ping L., "The Pattern Articulation Unit of Illiac III: Simulation Part I: Iterative Array, Transfer Memory, BOOLE Control", September 1968.

File No.

File No. 776	Borovec, Richard T., "The Pattern Articulation Unit of Illiac III; Display System: Design and Implementation, September 1968.

1.4.4 General Organization of the Pattern Articulation Unit

1.4.4.1 Iterative Array

The Iterative Array is a 1024-module parallel processor comprising nine horizontal planes of registers (see Figure 1.4.2; the nine planes of the IA are labeled M and P0 thru P7. These planes are imagined, for convenience, to be oriented horizontally and to be labelled by x,y coordinates. Vertical connectivity exists between cells in all planes having the same coordinates. In addition, each cell is interconnected with its eight nearest neighbors in a horizontal plane. These interconnections allow formation of logical functions of various combinations of vertical and nearest neighbor cells. The generic module, called a "stalactite", is shown in figure 1.4.4.1.

Furthermore, each plane of the IA is provided on each side with a row of border registers which may be thought to act as overflow bits.

The IA may be loaded and unloaded from main storage and may also read and write information into the Transfer Memory under control of PAU control. In this case, the M-plane is used as a buffer between the IA and TM.

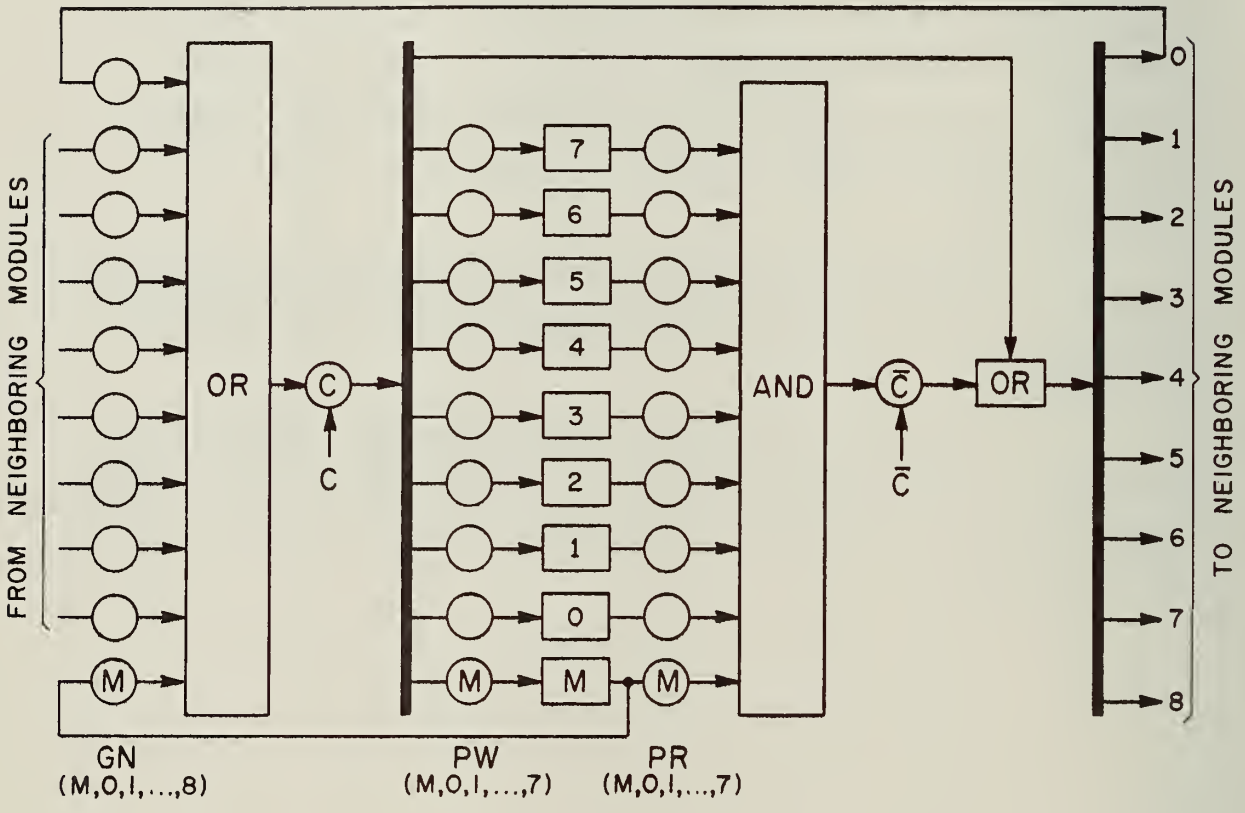


Figure 1.4.4.1 Generic Module, or "Stalactite", of the Iterative Array

1.4.4.2 Transfer Memory

The Transfer Memory (TM) is a random access core memory of 48 x 1024 (32 x 32) bits. Considered as a two-dimensional matrix (48 x 1024), the memory is arranged for read or write operations either by rows or columns.

The 48 (1024-bit) words are called long words, while the 1024 (48-bit) words are called short or z-words. The long words may only be loaded by means of the 1024-bit data path from the IA. The z-words may be loaded from main store under control of PAU control.

1.4.4.3 PAU Control

The PAU has its own control to enhance the parallel processing capabilities of Illiac III. PAU control decodes and executes all instructions sent from the TP and also controls the data flow both to and from the PAU. All data must pass through the control unit.

The control is of the stored-logic or microprogrammed type in order to facilitate the restructuring of the order code if the desirability of another instruction is evident.

Figure 1.4.4.3 illustrates the interconnection of the PAU and its control.

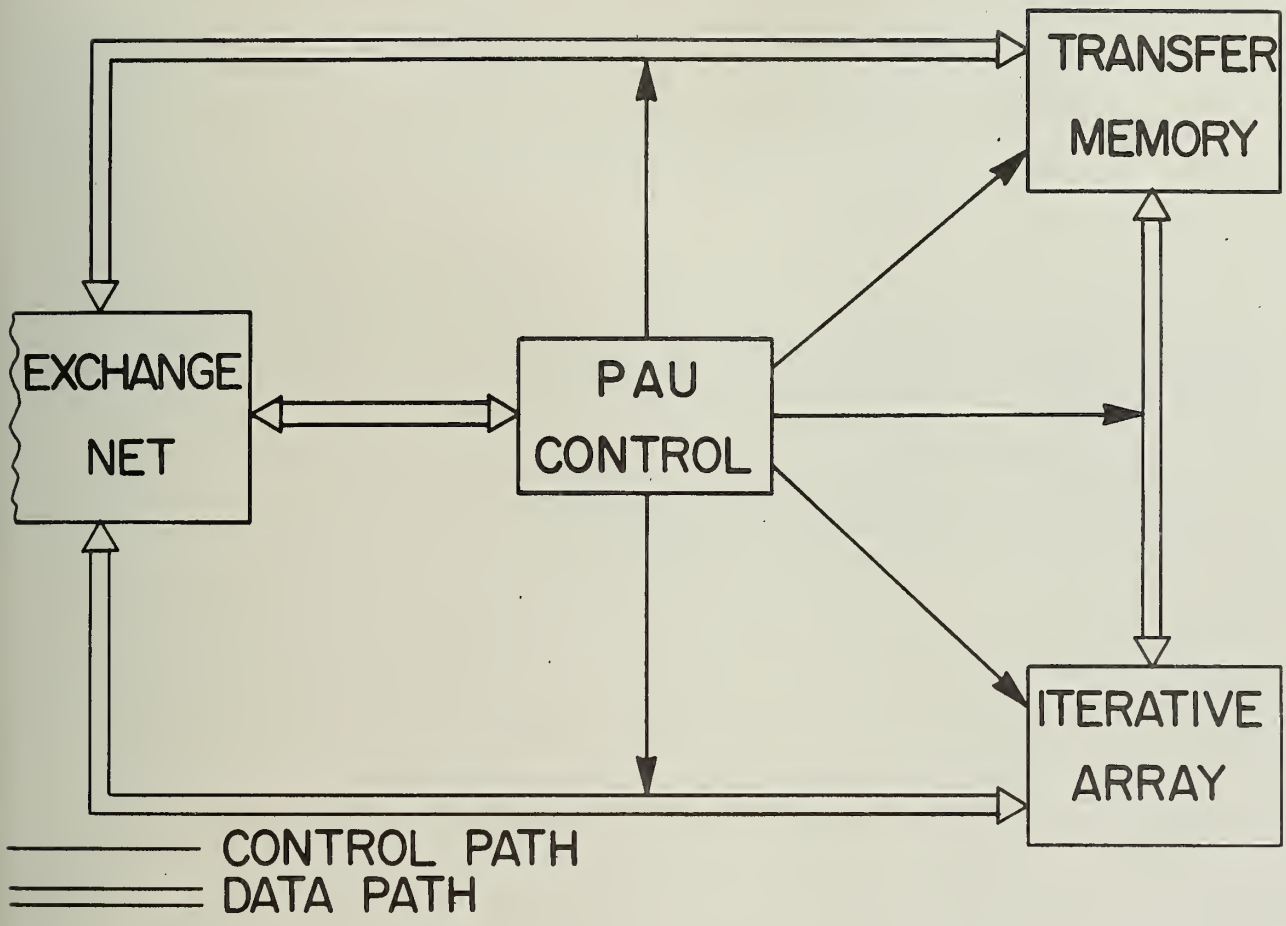


Figure 1.4.4.3 Block Diagram of PAU

1.5 Interrupt Unit

1.5.1 Function

The Interrupt Unit (IU) of Illiac III is a message store-and-forward processing unit through which all processor-processor communication takes place. In addition certain system-wide facilities, e.g. system accounting clock, etc., are maintained by the unit.

1.5.2 Documentation

Reports

Report No. 386

Goyal, Lakshmi N. and Lansford, Robert M., "Illiac III Computer System Manual: Interrupt Unit Volume 1: Logical Design", March 5, 1970.

Goyal, L. N., "The Hardware Realization of the Interrupt Unit of the Illiac III", Department of Computer Science, University of Illinois (in progress).

1.6 Exchange Net

1.6.1 Function

The Exchange Net is depicted in Figure 1 as having six Processor ports and eighteen Unit ports. The purpose of the Exchange Net is to provide a 50-bit Processor-to-Unit information path and a 50-bit Unit-to-Processor information path for every possible Processor-Unit pair.

1.6.2 Documentation

File Number

File No. 790	Krabbe, S. Paul, "A Discussion of Illiac III Processor-Unit Communiction Via the Exchange Net", March 8, 1969.

1.7 I/O Processors

1.7.1 Function

The Illiac III system contains two identical Input/Output Processors which directly control all I/O operations. Transfer of information between the main store and peripheral devices contemplated in the design includes communication with secondary storage media (e.g., magnetic disks and tapes, microimage stores), scan/display equipment and other computers.

Each Input/Out Processor (IOP) directly controls all input/output operations on the 8 channels under its jurisdiction. Each channel operates independently as a selector channel while in a multiplexing fashion the IØP directs channel operation in concert.

A channel in the Illiac III I/O system is the physical information transmission path linking an IOP to a set of I/O devices and their Device Controllers (DC's). The IOP is connected to a particular channel through a Channel Interface Unit (CIU). The CIU acts as a buffer between the very fast IOP and the relatively slow I/O devices. The CIU relieves the IOP of routine data packing/unpacking operations and response signal generation associated with data transfer.

The connection between a CIU and the peripheral devices/device controllers on the same channel is called the I/O Interface. This constitutes the main I/O data transmission path; it may connect very remote DC's to the computer. The I/O interface in the Illiac III System is a slightly extended version of the I/O Interface employed in the IBM System/360; this interface is fully compatible with the standard IBM System which has been widely adopted by I/O equipment manufacturers. Up to 256 independently addressable devices may be attached to each channel. At any time only one I/O device may be logically connected to the IOP.

Figure 1.7.1 shows a schematic of the I/O system.

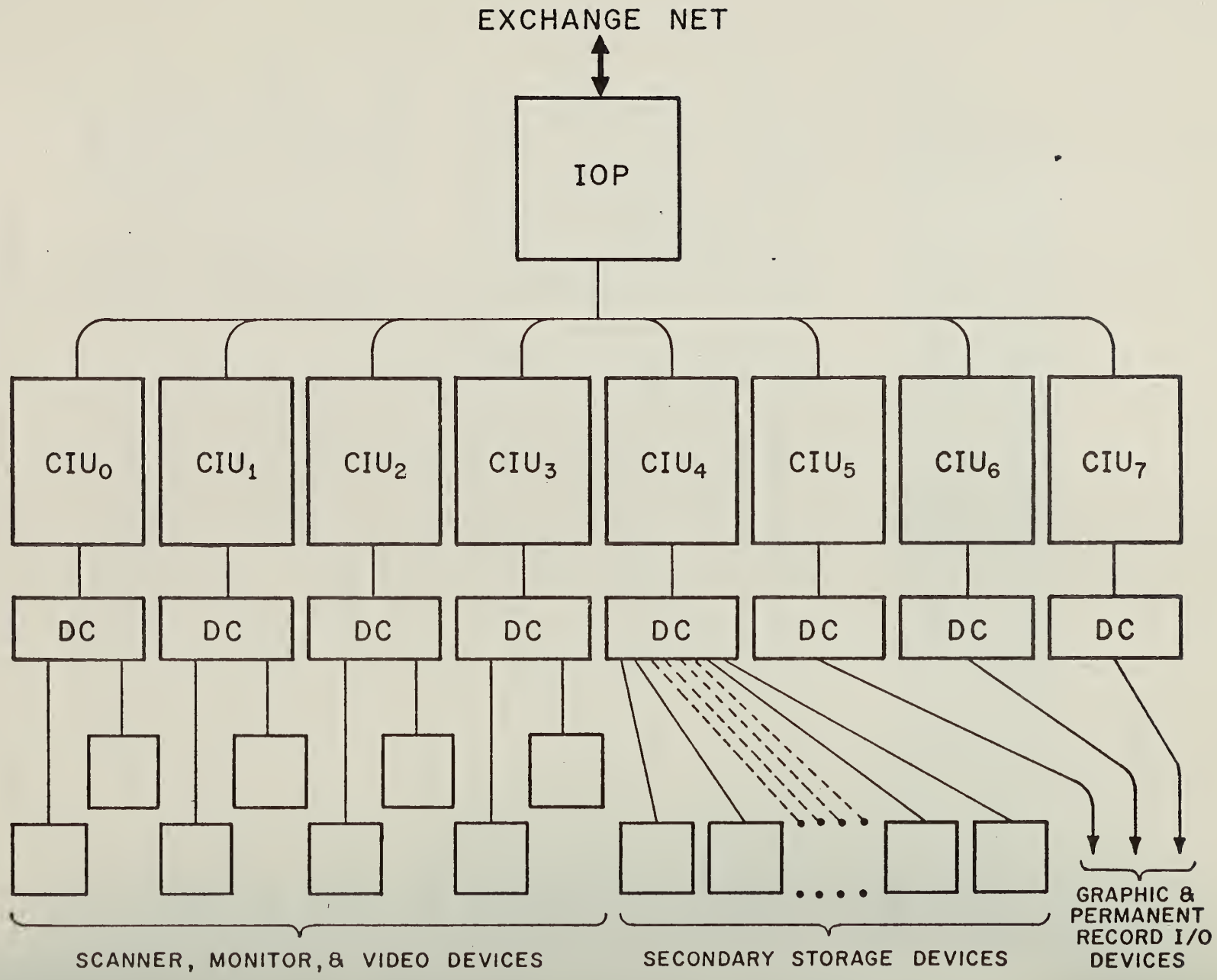


Figure 1.7.1 Schematic of I/O System

1.7.2 Design Features

The IOP is capable of controlling simultaneously, on a multiplex or time-shared basis, a large number of I/O devices with widely differing data rates. The basis of the multiplexing mechanism here is the concurrent execution of up to 8 independent I/O programs by the IOP. Each I/O program is defined by a sequence of I/O commands which, during program execution, are stored in main store. One program normally controls the data transfer over one channel, though auxiliary channels may also be involved.

The IØP design accommodates very high data rates (up to 1 megabyte per second is required for scanner data). Prior to the initiation of any data transmission command, the IØP determines whether or not sufficient channel capacity remains to insure that the maximum IØP data rate is not exceeded by adding this device's data to that already being handled. The IOP has available, as part of the I/O command to be executed, information concerning the data rates of all attached devices and the loading each imposes on the IOP. When the IOP is requested to connect a particular device to the I/O system, it compares the load which would be imposed by that device to the available IOP channel capacity. If it has sufficient unused capacity available to operate the device without transmission error, the request is accepted and the specified device is selected; otherwise the request is temporarily suppressed. Thus the IOP automatically ensures that its maximum channel capacity is never exceeded.

Data transfers may occur over all 8 channels simultaneously. Data transmission error, e.g., CIU buffer overflow, can occur if a channel is not serviced (i.e., supplied with or relieved of data) sufficiently fast by the IOP. Such a condition can occur in a low priority channel which is "locked-out" by one of higher priority. This type of error is largely avoided in Illiac III. For each device, the IOP internally detects when a data transfer may go critical. In this situation the corresponding channel is marked critical and is transferred to the position of highest priority on the IOP task list. Thus, it is normally serviced before a transmission error can occur.

1.7.3 Documentation

Report

B.H. McCormick, L.M. Katoh, V.G. Tareski, "Input/Output Processor: Registers and Control Flow," Department of Computer Science Report, University of Illinois, (in progress).

1.7.4 General Organization of the I/O Processor

1.7.4.1 The I/O Processor

Figure 1.7.4.1 shows the main registers and data paths of the IOP together with the various subprocessors governing them. The main "active" registers, associated with the Permuter, are the Interrupt Register (IR), the Logic Register (LR), the Distribution Register (DR) and the Arithmetic Register (AR). These are all 36 bits long (4 bytes of 8 bits and 1 flag each) and are used to hold data temporarily as various commands are being processed. None of these are directly accessible to the programmer.*

The main "storage" registers are divided into three groups: (1) System registers (11 bytes), (2) Program Module registers (maximum of 8 modules, each of 8 bytes) and (3) Channel Module registers (maximum of 8 modules, each of 26 bytes). These registers contain data which may be used by the I/O Supervisor to guide the transmission of data. With the exception of all base registers (other than that of the Segment Table) they are all accessible by programming at appropriate stages in the data transmission (e.g., at an abnormal/normal termination of data transfer, etc.).

In connection with Figure 1.7.4.1 it should be noted that the output from the Permuter, the Distribution Bus (DB), is not a register. It consists of 36 lines which serve as inputs to nearly all of the registers and control sections of the IOP. The DB is the main avenue of information transfer internal to the IOP.

Finally the role of the IOP as a way-station between the peripheral devices (scanners, disks and other secondary storage devices, etc.) and the Exchange Net is exhibited in Figure 1.7.4.1 by the CIU/IOP Interface. This interface consists of a set of data and control lines which connect (a maximum of) eight Channel Interface Units (CIU's) to their common IOP. The data lines here are organized so that in general no reformating of the data is required and direct data transmission between the appropriate CIU and the Exchange Net is achieved.

---

*Information transfer between the IOP and the Exchange Net via the Permuter and its associated temporary registers follows a close parallel to that in the Taxicrinic Processor (see Section 1.1.4.6).

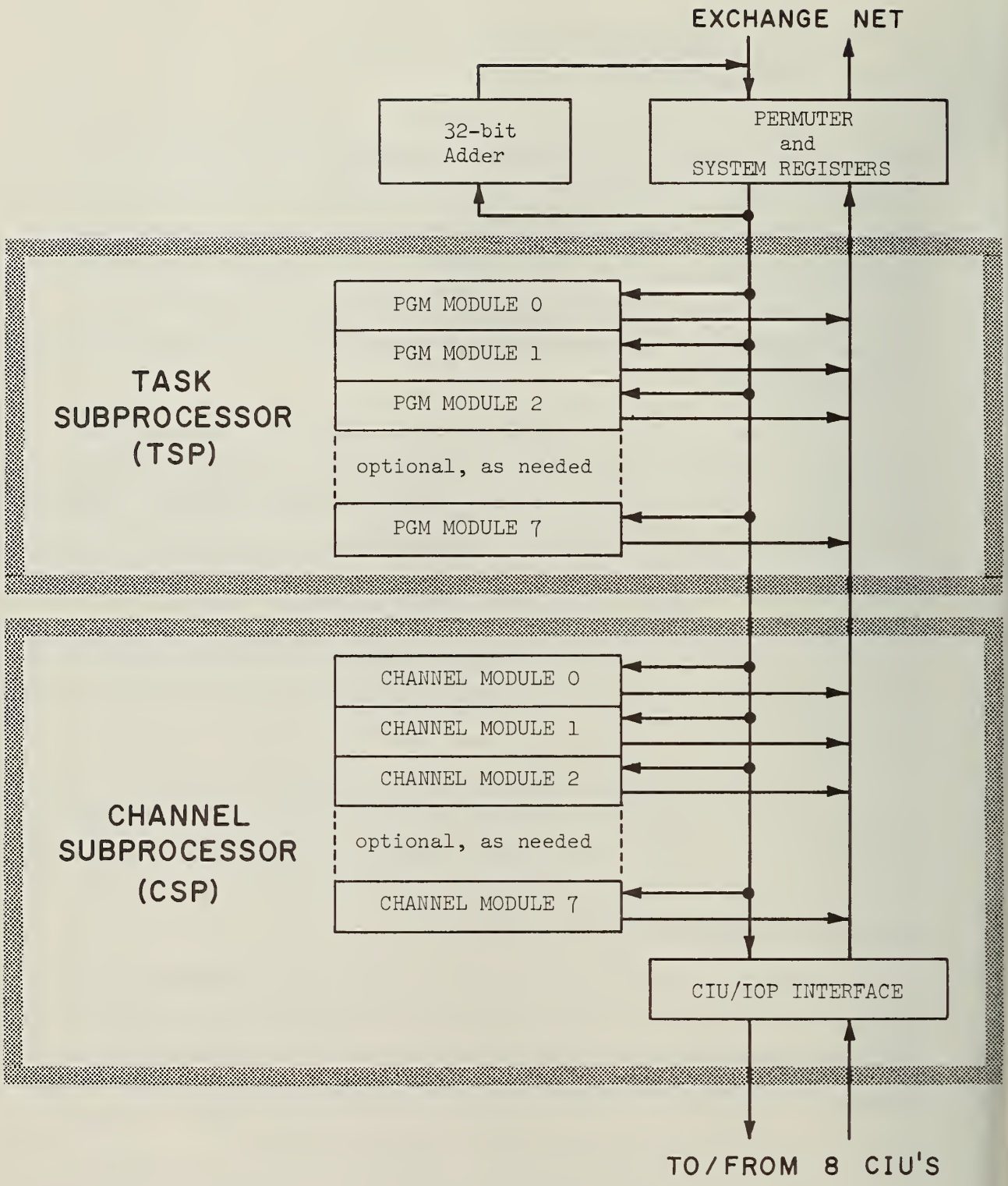


Figure 1.7.4.1 Main Registers and Data Paths of the I/O Processor

1.7.4.2 The IOP Subprocessors

The IOP control is divided into two parts: the Task Subprocessor (TSP) and the Channel Subprocessor (CSP). The Task Subprocessor controls the sequencing of tasks within the I/O Processor, the handling of the various interrupts, and the execution of device independent commands (DIC's). The Channel Subprocessor handles the execution of device dependent commands (DDC's); thus it controls the information exchange with the Channel Interface Units (CIU's) and initiates memory access requests when data transfer is required.

These subprocessors in turn communicate with the Program/Channel Modules of the IOP by means of "control lines" which give commands to the program/channel modules and "output lines" which give the two basic subprocessors information about the status of the program/channel modules. There is one Program Module for each independent program (up to a maximum of 8) that the IOP can simultaneously execute in a multiplexing mode. Similarly there is one Channel Module for each channel attached to the IOP (up to a maximum of 8). These program/channel modules are in general fairly independent and need only receive a small number of control lines to perform rather complex operations.

The main register groups of the IOP are the Permuter (and associated registers IR, LR, DR and AR), Program Module registers, Channel Module registers, various System Registers (3 Base Registers and Residual Capacity Register), the 32-bit adder and the CIU/IOP Interface.

The Permuter is a system of gates and drivers which is central to the transfer of information in the IOP. Its role and design are identical to the Permuter in the Taxicrinic Processor. Discussion of the other register groups will be deferred. (See Sections 1.7.4.3 — 1.7.4.7 for detailed descriptions.)

The 32-bit Adder in the I/O Processor is used to perform binary addition (primarily for address construction) with the TP. It is also used to generate the outputs of masking operations (in DIC's). The

adder uses 2's complement number representation and employs two levels of carry lookahead to hasten carry propagation. The 32-bit Adder is identical to that used in the TP (see Section 1.1.4.8).

Inputs to the adder are obtained from the Distribution Register (DR) and the Permuter Distribution Bus (DB). These two inputs can be gated to the adder only in "true" form. To calculate the difference of two numbers a similar strategem to that used by the TP is used. In all respects the Permuter-Adder complex performs as described previously (Section 1.1.4.8).

The principal uses of the CIU/IOP interface (see Figure 1.7.4.1) between the Channel Subprocessor (CSP) and the Channel Interface Unit (CIU) are to convey control information associated with the initiation and termination of channel operations and to transfer data between the CIU and a CSP. This interface consists of a set of data and control lines which connect (a maximum) of eight CIU's to their common CSP. These lines are in two groups--those which are shared in common by all eight CIU's and those which are separate for each CIU. Design of the interface was predicated upon minimizing the number of the latter lines. Further description of this interface is deferred until Section 1.7.4.7.

1.7.4.3 Base Registers

Three Base Registers (BR's) play a particularly prominent role in the memory accessing scheme to obtain information from the segments of the current process. These are the base registers of the Segment Table, the Command Segment and the Descriptor Segment.

In addition two base registers are associated with each Channel Module. These latter are described in Section 1.7.4.5.3.

1.7.4.3.1 Segment Table Base Register

All segments employed by the IOP in the execution of its 8 independent programs, including the Command Segment and the Descriptor Segment defined below, are identified by Segment Table entries.

Both the IOP and the TP use a common Segment Table format. (For an extensive discussion of Segment Table techniques, see Chapter 5.) Accordingly the key to the IOP addressing system is the Segment Table Base Register (STBR). The STBR is three bytes long, and in format and use plays a role similar to the BR#0 of the TP. (See Section 1.1.4.3.)

1.7.4.3.2 Command Base Register

The Command Base Register, CBR, defines and locates the Command Segment which contains all the I/O programs. The command segment is a contiguous, read only segment with its size determined by the bounds byte of the CBR. The CBR is three bytes long. The first byte is a bounds byte, the last halfword is a page address and the flags are an access code (usually set at 10, to indicate a contiguous, read-only segment).

1.7.4.3.3 Descriptor Base Register

The Descriptor Base Register, DBR, defines and locates the descriptor segment which contains all the I/O descriptor lists. The descriptor segment is a contiguous, read-write segment with its size determined by the bounds byte of the DBR. The DBR is three bytes long: the first byte is a bounds byte, the last halfword is a page address and the flags are an access code (usually set at 11, to indicate a contiguous, read-write segment).

1.7.4.4 Program Modules

To enable the IOP to execute its programs independently, it contains eight (maximum) Program Modules (PM's), each of which contains all necessary information for the execution of a particular program. Those modules are named PM_0 to PM_7; the assignment of a particular program to a PM in the IOP is the duty of the I/O Supervisor. Sequencing of the commands within the Program Module is then the nominal province of the Task Subprocessor (TSP), see Section 1.7.4.2

Each PM contains three fast registers: the Program Status Register (halfword); the Command Pointer Register (halfword), which stores the address of the current (or next) program command; and the Command Register (word), which stores the current program command.

All command addressing is done relative to the Command Base Register (Section 1.7.4.3.2), which is shared in common by all Program Modules.

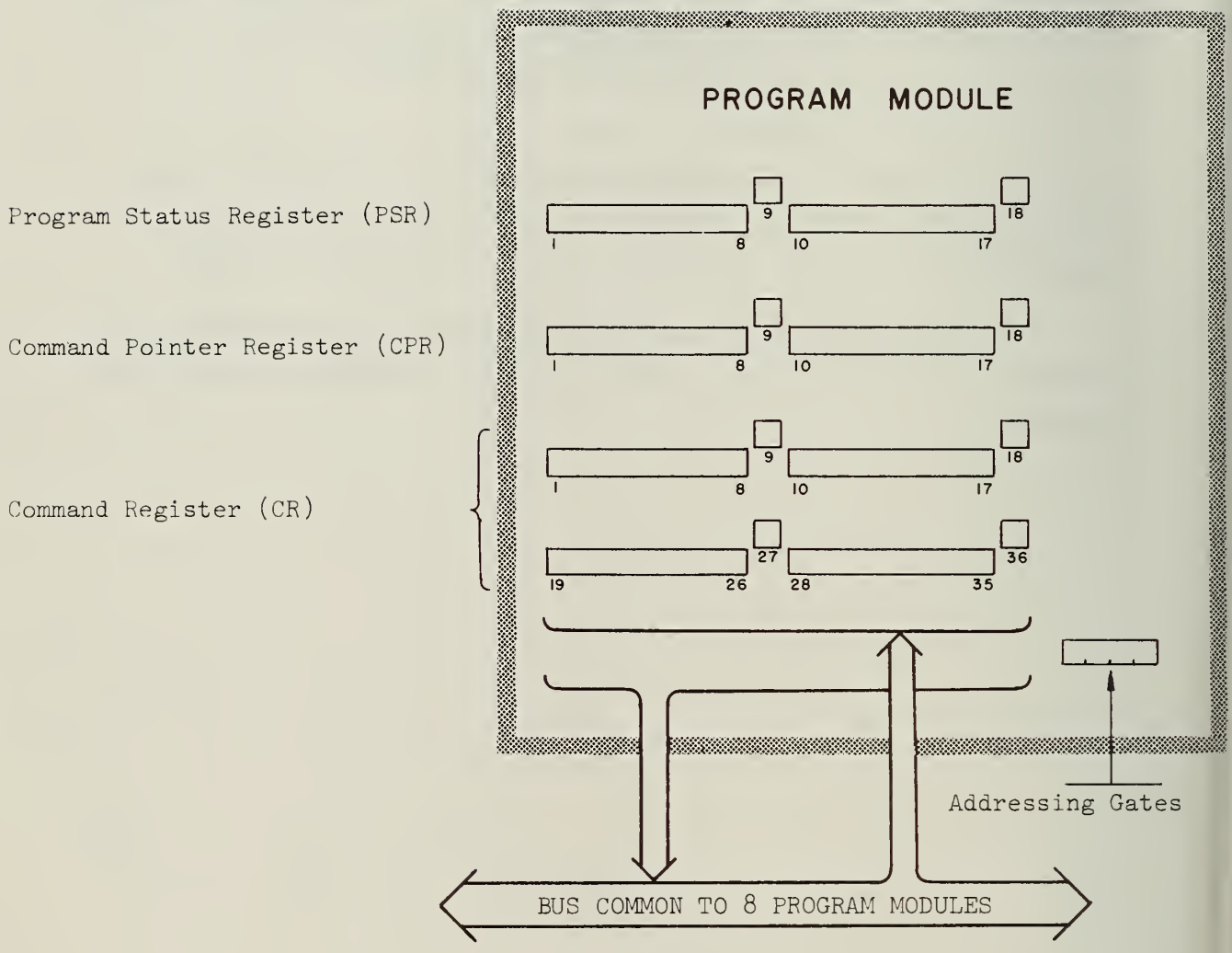


Figure 1.7.4.4 Schematic of Program Module

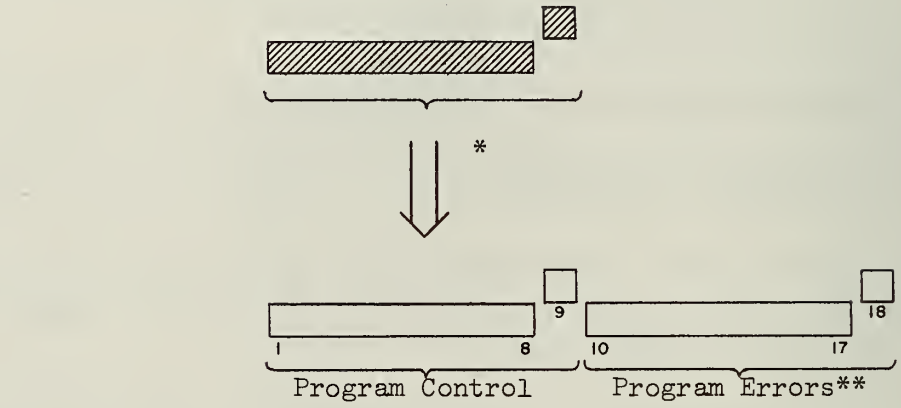
1.7.4.4.1 Program Status Registers

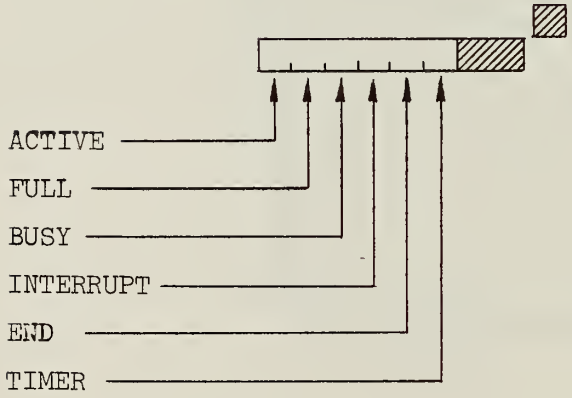
Sequence control in the execution of the I/O program is maintained through the use of a Program Status Register (halfword). Contents of this register comprises the Program Control Byte (bits 1-9), and the Program Errors Byte (bits 10-18).

The Program Control Byte is not accessible by the programmer; the format of the byte is however described in Figure 1.7.4.4.1. (For further explanation of the significance of these bits see the IOP Manual.)

The Program Errors Byte, however, is preserved, upon program termination, in the Program-end Status Descriptor (see Section 3.2.2.2.2.1 for a description of this byte.)

Corresponding to the eight (maximum) Program Modules there are eight Program Status Registers (PSR's), designated PSR_0 - PSR_7. Each of the eight program status registers is assigned to one of the eight (maximum) concurrent I/O programs.





* Program Control byte is suppressed prior to storing Program Status in the Program-end Status Descriptor.

** See Section 3.2.2.2.2.1

Figure 1.7.4.4.1 Program Status Register

1.7.4.4.2 Command Pointer Registers

The Command Pointer Register, CPR, in conjunction with the Command Base Register, addresses the sequences of commands in main store which define the I/O process.

Corresponding to the eight (maximum) Program Modules there are eight CPR's, designated CPR_0 - CPR_7; each is assigned to the associated Command Registers. Each CPR is one halfword long and contains a 16-bit pointer and flag bits.

Command Pointer Registers are loaded from Start I/O Interrupt Commands and are stored in the End Status and Interrupt Acknowledgment. They are incremented immediately after the "command fetch" and are sometimes adjusted during command execution.

1.7.4.4.3 Command Registers

Each Program Module stores the current command of its active I/O program in its Command Registers (CR). This register is one word long and each may store either a Device Independent Command or a Device Dependent Command.

Corresponding to the eight (maximum) Program Modules there are eight CR's, designated CR_0 - CR_7. Each of the eight command registers CR_0 - CR_7 is assigned to one of the eight (maximum) concurrent I/O programs. Command Registers are loaded during the "command fetch" sequence and are stored in the Program-end Status Descriptor.

1.7.4.5 Channel Modules

A channel is defined as a physical information transmission path (data and control lines) linking an IOP to a group of peripheral devices. Each channel is connected to the IOP via a Channel Interface Unit (CIU). The CIU acts as a buffer between the very fast IOP and the relatively slow peripheral devices. The CIU relieves the IOP of the routine byte packing/unpacking operations and response signal generation associated with data transfer.

Associated with each (of the maximum eight) channels is a Channel Module, which contains all information necessary to execute a device dependent command; thus it controls the information exchange with the corresponding Channel Interface Unit (CIU) and initiates memory access requests when data transfer is required. Sequencing of data transfers dictated by the Channel Module is then the nominal province of the Channel Subprocessor (CSP), see Section 1.7.4.2.

For each of the 8 channel modules there is a Channel Status Register (CSR) which contains, as part of the status, the name of the program module (KEY) which is currently controlling this channel. The program key can only be loaded under direction of the I/O Supervisor, (e.g. by a START IO command) since the channels are a system-wide resource.

In addition each channel module contains two data descriptors (doubleword each), referred to as the Descriptor and the Shadow Descriptor respectively. Also associated with each descriptor is a base register (3 bytes each).

Figure 1.7.4.5 gives the format of a Channel Module.

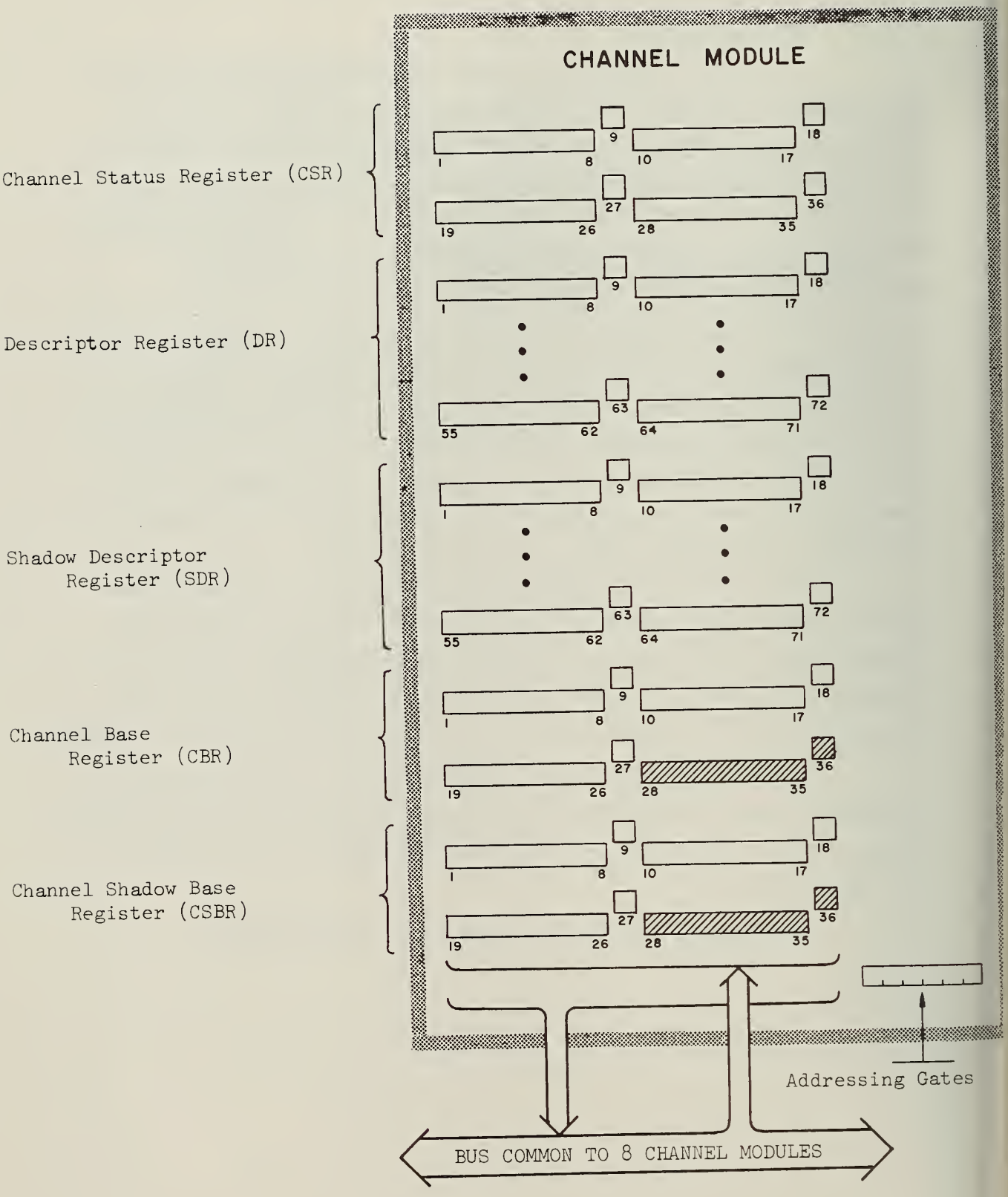


Figure 1.7.4.5 Schematic of Channel Modules

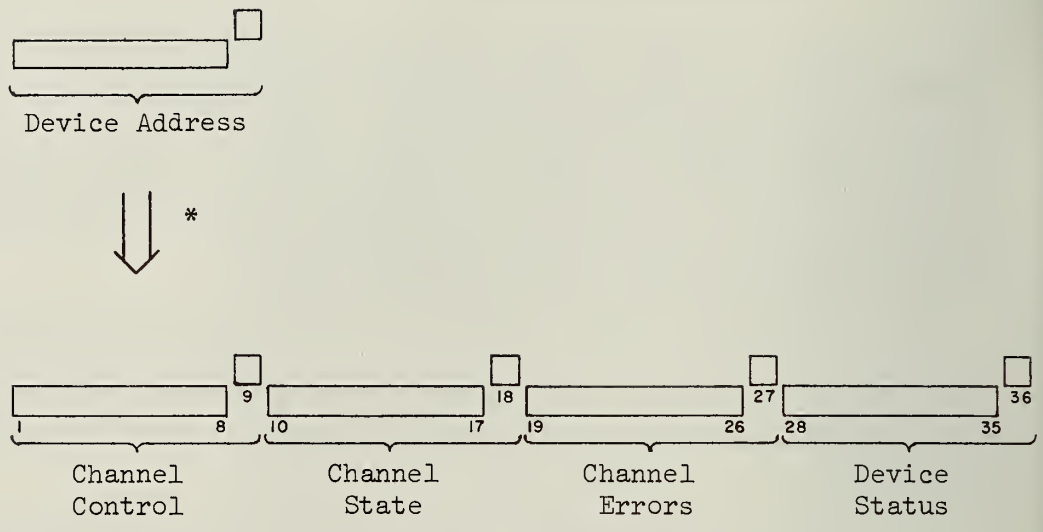
1.7.4.5.1 Channel Status Register

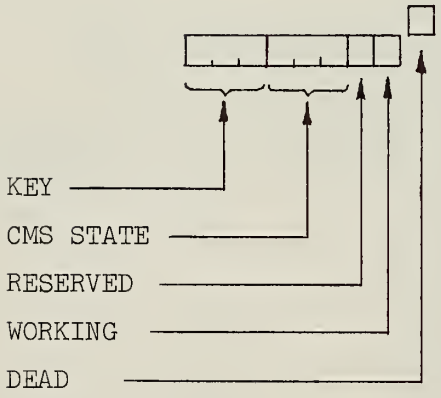
Sequence control in the execution of a device dependent command (i.e., involving data transfer) is maintained through the use of a Channel Status Register (word). Contents of this register comprises the Channel Control byte (bits 1-9), the Channel State byte (bits 10-18), the Channel Errors byte (bits 19-27) and the Device Status byte (bits 28-36), see Figure 1.7.4.5.1.

The Channel Control byte is not accessible by the programmer; the format of the byte is however described in Figure 1.7.4.5.1. (For further explanation of the significance of these bits see the IOP Manual.)

The remaining three bytes (Channel State, Channel Errors and Device Status), however, are preserved, upon termination of data transfer, in the Channel-end Status Descriptor. (See Section 3.2.2.2.1 for a description of these bytes.)

Corresponding to the eight (maximum) Channel Modules there are eight Channel Status Registers (CSR's), designated CSR_0 - CSR_7. Each of the eight channel status registers is assigned to the correspondingly numbered channel.





* Device Address byte replaces Channel Control byte when the Channel Status Register is stored in the Channel-end Status Descriptor (see Section 3.2.2.2.1).

Figure 1.7.4.5.1 Channel Status Register

1.7.4.5.2 Descriptor and Shadow Descriptor Registers

A Data Descriptor can be considered as an "extended pointer" wherein the first three halfwords have the pointer stack format (Section 1.1.4.4.1 on the TP) and the last halfword is the Byte Count (or field size). For a review of this format and its interpretation, see Figure 3.2.2.2.1.

The Descriptor Link field, in conjunction with the Descriptor Base Register, addresses the next descriptor in main store. Each link contains a 16-bit pointer and a 2-bit flag code which identifies the current descriptor type. The link here plays a role in the definition of the descriptor list completely analogous to that it previously served as the defining mechanism for pointer stacks.

The descriptor link contents are initially computed upon the execution of a Start I/O command and finally are stored in the Program-end Status Descriptor. During DDC execution links are loaded sequentially from all descriptors accessed; they are sometimes also adjusted during command execution.

The Descriptor Value and Name fields retain the same interpretation they have for pointers generally.

The Byte Count indicates during data transmission the number of data bytes remaining to be transferred. This portion of the Descriptor Register is a halfword long and contain 16-bit residual byte count and two 1-bit flag variants. Byte Counts are loaded from Data Descriptors and stored into Channel-end Status Descriptors. They are checked and decremented during data transmissions.

Each Channel Module contains a (doubleword) Descriptor Register and a (doubleword) Shadow Descriptor Register. These are designated DR0, ..., DR7 and DSR0, ..., DSR7 respectively for the eight Channel Modules. Descriptor and Shadow Descriptor Registers have identical format and field interpretation.

The Shadow Descriptor provides temporary storage for a channel module searching a new data descriptor while the first descriptor is still active. This look-ahead scheme, using a Shadow Descriptor, facilitates rapid data chaining. The register configuration of the Shadow Descriptor is in no way distinct from the Descriptor Register normally associated with a channel.

The doubleword Shadow Descriptor Register (i.e. Link, Value, Name and Byte Count) is loaded from a data-chained Data Descriptor. The Byte Count in the descriptor register may be decremented during data transmission operations.

1.7.4.5.3 Channel Base Registers

Two base registers are assigned to each channel. These registers are associated with the Descriptor and the Shadow Descriptor, respectively, of the channel. The base registers are accordingly designated CBR0 through CBR7 for the descriptor base registers, and CSBR0 through CSBR7 for the shadow descriptor base registers of Channel Modules 0 through 7 respectively.

These registers are not accessible to the programmer, even by an interrupt command--thereby insuring the integrity of memory. They can only be hardware loaded from the Segment Table upon the first reference to the corresponding descriptor, in particular as dictated by the name field (halfword) of the descriptor. If the descriptor is visualized as a pointer (link, value and name) augmented by a field size (byte count), it is then seen that the CBR and CSBR refill mechanism is a close parallel to that employed in the Taxicrinic Processor.

1.7.4.6 CIU/IOP Interface

The principal uses of the interface between the Channel Subprocessor (CSP) and the Channel Interface Unit (CIU) are to convey control information associated with the initiation and termination of channel operations and to transfer data between the CSP and a CIU. This interface consists of a set of data and control lines which connect eight CIU's to their CSP. These lines are in two groups--those which are shared in common by all eight CIU's and those which are separate for each CIU.

The shared lines are the two data buses, bus to and bus from, and the two control buses, control to and control from. The data buses carry all of the data between the CSP and the CIU's while the control buses regulate data flow and channel operations.

The separate lines are used to establish the connection between a particular CIU and the CSP. These lines are ready to, ready from, critical from, and reply from.

Figure 1.7.4.6 shows schematicably the CIU/IOP interface. The interface signal names are named with respect to the CIU, that is, to signal names mean a signal to the CIU from the CSP while from signal names denote a signal originating from the CIU and going to the CSP. (The convention for the CIU/Device Controller (DC) interface is also with respect to the CIU, namely, in means signals coming in to the CIU from the DC and out means going out of the CIU to the DC.)

Figure 1.7.4.6 CIU/IOP Interface

1.7.5 Memory Access

The main task of the Memory Access Sequence in the IOP is to construct the absolute address of the memory location(s) desired. The access mechanisms available are identical to those of the Taxicrinic Processor: both contiguous and partitioned data access are permitted, as specified by the appropriate base register. Also the access privilege checks are handled in the same way.

In practice, however, the system programmer is ill-advised to use partitioned mode accessing unless all transfers currently handled by the IOP are restricted to low or medium data rate devices. The use of partitioned mode automatically imposes a 2-memory cycle time delay and effectively reduces integrated channel capacity by approximately 50%. However the option remains as a challenge to the skilled systems programmer, and can be used to great advantage in dealing with paged low speed terminals.

1.7.5.1 Address Bounds Check

The address bounds check is performed on all outgoing memory addresses to insure that the constructed address falls within the area allotted to the segment in question. The method is the same as the TP's and the description and figure 1.1.5.2 should be examined again. An address bounds failure in the IOP causes termination of the I/O program.

1.7.5.2 Access Privilege Check

For file security the access control bits identify the types of access permitted to each segment. The full access mode of the TP is described in Section 1.1.5.3. The rightmost two flags of a base register are used for the access code. The three system base registers expect to receive the following access code assignments:

Segment Table Base Register	10	(read only, no write)
Command Base Register	10	(read only, no write)
Descriptor Base Register	10	(read only, no write)

Any other assignments that are detected during the execution of a Load Base Register command cause the Acknowledgment to contain a failure flag.

1.8 Channel Interface Unit

1.8.1 Function

A Channel Interface Unit (CIU) provides 3 words of buffering between the very fast I/Ø Processor and the relatively slow peripheral devices. In conjunction with this buffering function the CIU provides the necessary interface control signals to the device controller for all data transfers, and it properly formats this data as it flows in and out of its buffers. The CIU also insures that the control signals to the device controller occur in the proper time sequences, as defined in the IBM System/360 Interface Manual.

1.8.2 Documentation

Report

Wenta, J.V., "Channel Interface Unit Manual," Department of Computer Science Report, University of Illinois, (in progress).

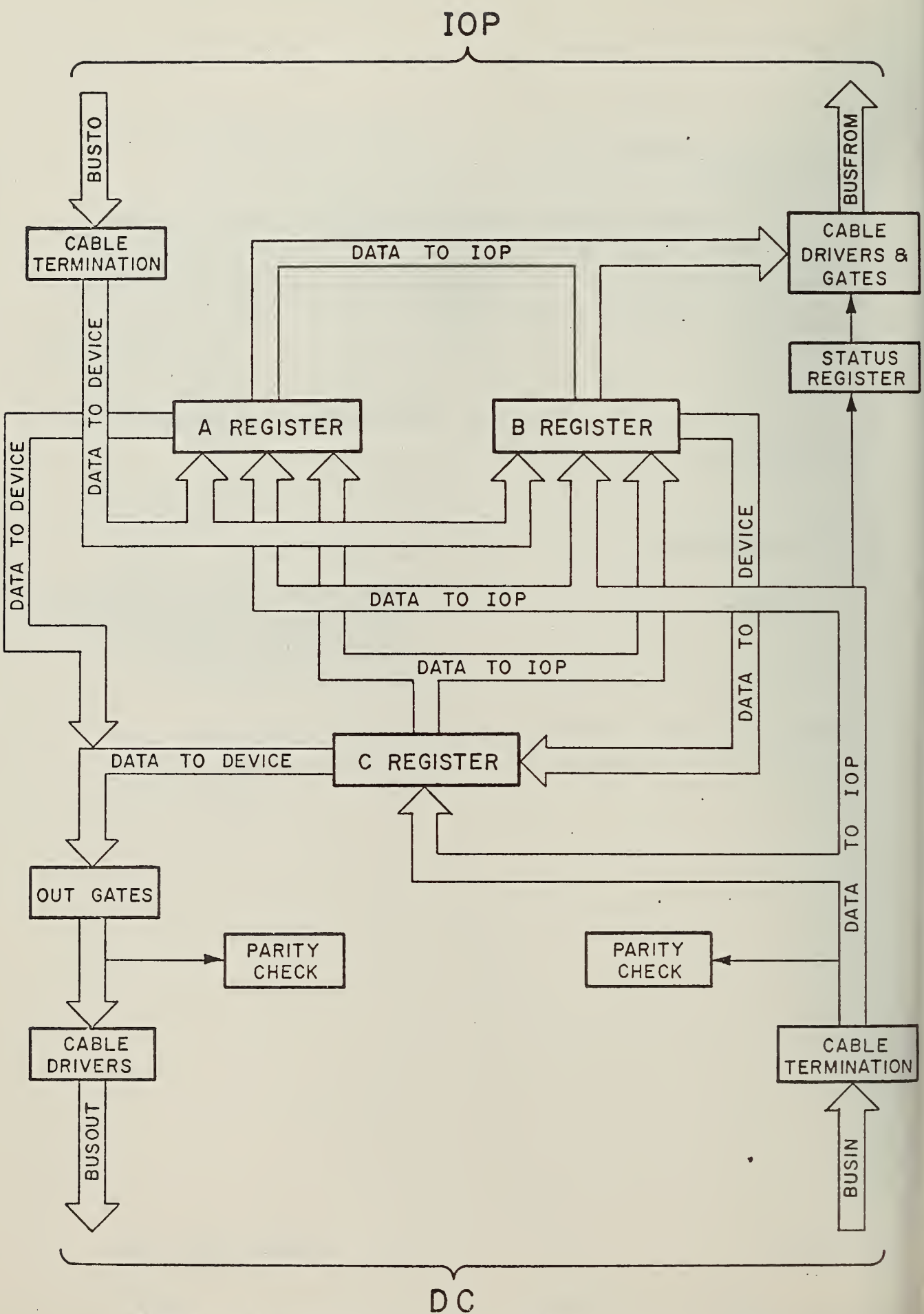


Figure 1.8.1 Registers and Data Paths of a Channel Interface Unit

1.9 Device Controller

1.9.1 Function

A Device Controller (DC) provides the proper interface control signals between the peripheral devices and the I/Ø channel. With respect to the channel the Device Controller affords the standard IBM System/360 Interface. With respect to the peripheral device the Device Controller assumes the task of decoding the various control signal sequences, but enough flexibility is allowed to enable the peripheral device to control such DC features as mode of operation (multiplex or burst) and the suppressibility of device data.

1.10 Secondary Storage

The I/O system design of Illiac III has been predicated upon the use of commercial disks and tapes. For budgetary reasons however, the machine at present is provided with no secondary storage.

One intrinsic difficulty arises in the use of commercial storage devices: the internal use of a nine-bit byte in Illiac III, i.e. allowing in each byte a flag bit with every 8 data bits. Two strategies are adhered to in transferring to/from an 8-bit secondary store:

1) Ignore the 9th bit, as used in communicating with the 8-bit organization of the IBM system 360/75. All I/O interfaces are compatible with IBM standards. (This normally entails some reformatting of the data)
2) Pack/unpack every eight flags into a byte which is then transferred as part of the data stream. This conversion can be done in appropriately modified Channel Interface Units. Transmission must be by double-word, starting on a natural double-word boundary.

Alternatively, we can exploit the bit-serial design of the storage device itself (e.g., disks), so that device indexing by 9-bit bytes is retained internal to the storage device itself. With integrated circuitry this change-over is quite reasonable.

Tertiary storage devices (photomemories) are now becoming commercially available. Trillion-bit stores are of considerable interest in mammouth image processing applications, e.g. bubble chamber data analysis and brain mapping. The card indexing mechanisms of these tertiary stores are also being adapted to provide rapid image access. Through the closed circuit television cameras of Illiac III, these micro-images could be readily encoded for transmission to the central computer. This latter development was foreseen, and the Illiac III is well adapted to exploit this potentiality.

1.11 Scanner-Monitor-Video System

1.11.1 Function

Facilities for the acquisition of pictorial information are basic to the mission of the Illiac III computer. In addition there is also a need for interactive display systems to present both intermediate and final processed data. The subject of graphic display terminals has been extensively discussed in the literature. Scan-display systems oriented towards image processing, however, with particular attention to bypassing the central processing unit for as many tasks as possible, have not had comparable development. The system described below, called the 'Scanner-Monitor-Video' (SMV) system of Illiac III, is focused on this latter area. Central to this system is the S-M-V Controller which serves to interdigitate all modes of scan/display.

A further new direction in image acquisition and display is to append a Video Communications Net to the central computer so as to provide both local and remote users with video communication to a centralized image processing facility. Design of the SMV system has been predicated upon this latter development.

1.11.2 Design Features

Figure 1.11.2 shows the system as it has evolved. High resolution scanners allow the scanning of the film for metrological and monitoring purposes and also allow the construction of images on film. Incrementally-driven high resolution monitors are slaved to the scanner system in a manner which allows the monitors to share inexpensively the S-M-V Control.

The video communications net provides closed circuit television facilities for both local and remote users. Signal routing is achieved through a manually-operated video switching matrix. Video images can be treated as if they were on film and thus the same encoding techniques and the same programs can be exploited for both without program change.

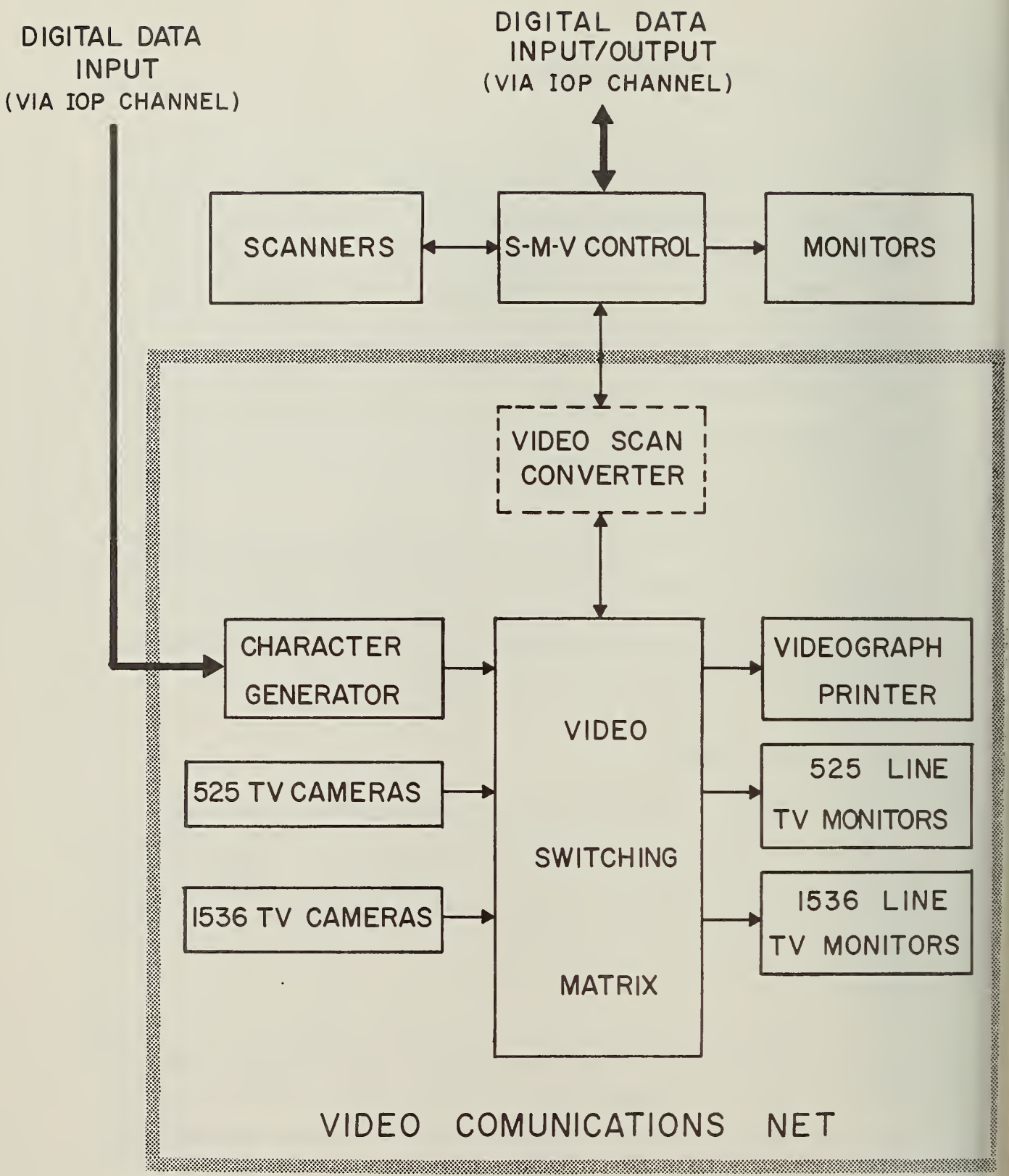


Figure 1.11.2 - Block Diagram of the Scan/Display System. Note that the Video Scan Converter can be bypassed under program control.

Both 1536 and 525-line CCTV cameras are available for image encoding and acquisition. To monitor high resolution image encoding, video consoles, consisting of two 1536-line TV monitors and a teletype set, provide information display at the user's end. The Videograph printer outputs a fascimile copy at the rate of one sheet every 0.8 seconds, where the copy can have any admixture of text, graph or half-tone pictures. The character generator provides facilities for message handling and display in general, particularly for computer-aided instruction, which along with the scanner provides the flexibility of displaying line drawings and half tone pictures intermixed with the text. More details about the devices shown in Figure 1.11.2 are given in Sections 1.11.4 on.

1.11.3 Documentation

Publication	Dunn, L.A., et.al., "Parametric Description of a Scan-Display System", AFIPS Conference Proceedings, Vol. 34, p. 187-206.
Report	
Report No. 320	Divilbiss, J.L., "Illiac III Scanner Analog Circuits", April 10, 1969.
File No.	
File No. 794	Dunn, L.A., et.al., "Programmed Aids for the Scan-Display System of Illiac III", (to be used with Report No. 308), March 14, 1969.
File No. 810	Franco, Sergio, "A High Speed Analog Divider for the Illiac III Scan System", October 3, 1969.
File No. 805	Franco, Sergio, "Dynamic Focusing and Pincushion Correction Circuit for the Illiac III Display System", August 12, 1969.
	Videograph System Manual, Model 9900-2 (U/D) Display Control Unit 990 Series (1 volume).
	Ampex Videofile 6205457, Sync Generator Manual, Service Manual, January 15, 1969 (1 volume) (Univ. of Illinois).
	Ampex Videofile 6205454, Slow Scan Monitor Electronics Unit Service Manual, January 15, 1969 (1 volume) (Univ. of Illinois).
	Ampex Videofile 6204487-01, VF-100 Camera Head and Camera Control Unit Service Manual, January 15, 1969 (1 volume) (Univ. of Illinois).
	Ampex Videofile, Camera Head and Control Unit, Schematic Diagrams and Parts Lists January 15, 1969 (1 volume) (Univ. of Ill.).
	A.B. Dick Videograph, Service Manual Page Printer Model 9020 (1 volume) (Univ. of Illinois).

A.B. Dick Videograph, Schematics and Parts Lists, Page Printer Model 9020 (1 Volume) (Univ. of Illinois).

1.11.4 Scan-Display Devices

1.11.4.1 Scanners

The scanners are flying spot scanning systems with an added diquadrupole coil for astigmatic defocusing of the spot into a line element to achieve a slit mode. All scanners are capable of either scanning from developed film or photographing onto unexposed film. The optical path of the beam is split, with one path transversing the film and the other path through a reference grid to establish stability against engraved fiducial marks.

Several types of media transports are provided to handle the projected range of problems. A 70 mm. scanner is provided primarily for long-format 70 mm. film. Here the format of the raster is 2.362 inches x 3.522 inches, and the minimum spot size is approximately 0.001 inch at the film. Due to the length of the frame to be scanned, scanning is done in two steps. The two horizontal halves of the frame are scanned successively with a 4 mm. overlap to establish half-frame continuity. Large motors are used for slew and gross positioning of the film and a small digital stepper motor is used for fine positioning of the frame. Frame position sensing is accomplished by using the digital stepper motor as a tachometer and by counting sprocket holes. Total film capacity is 1000 feet.

A scanner for handling 47 mm. film is similar to the 70 mm. transport design except for the following:

The film format is different. A friction drive is used on the digital stepper motor, since the film is unsprocketed. The frame position is determined by sensing small index blocks at the lower edge of the film using a fibre-optics light guide and a photodiode.

The microform scanner contains three units. The first is a 35 mm. full frame digitally controlled camera/scanner which can read light through the film, or alternatively record computer-generated images. The second unit contains a 16 mm. Bolex camera for making computer-generated black and white movies and a modified 16 mm. film editor for scanning 16 mm. film of all types. The third unit is a microfiche transport mechanism for scanning and producing a single microfiche in the 72 image COSATI format. For the three different units the C.R.T. raster is adjusted optically to fit the particular frame size.

A fourth type of scanner is built around a microscope with a digitally controlled automatic stage. Positional accuracies are on the order of $\pm$ 2 microns, and the maximum slide area coverage is 1.2 inches x 1.2 inches. Variable reduction is available from a four objective rotating turret. Full visual observation is available to an operator.

1.11.4.2 Monitors

The monitors consist of 21 inch cathode ray tubes controlled in a manner similar to the scanner C.R.T.'s; viz., digital position counters control the spot location through accurate, high-speed digital-to-analog converters. The monitor counters are digitally controlled directly from the S-M-V Controller via an incremental communications scheme; essentially the only commands issued by the S-M-V Controller for the monitors are increment the counters, decrement the counters, reset the counters, and reset the parameters. Therefore, any spot movement possible on a scanner C.R.T. can be accomplished on the monitor C.R.T. The video input for the C.R.T. grid is also synchronized by the S-M-V controller.

Included with the monitors for communications to a central processing system are a selectric typewriter, microtape input/output tape drives, and a light pen for cursor control.

1.11.4.3 Video Switching Matrix

The video switching matrix is a mechanical switching network comprising eight single-pole, six position coaxial rotary switches. The arrangement allows any of eight sources (high resolution TV, output of character generator or SMV) to drive one or more destinations (input of Character Generator, SMV Video Digitizer, Videograph Printer, or video Consoles). A destination may be driven by only one source, while a source may drive several destinations. Switching time is on the order of one second.

1.11.4.4 Character Generator

The Character Generator is designed to accept up to 512 ASCII characters into its 4096 bit memory. A 9 x 11 dot matrix is used to develop each character into the appropriate video levels. The characters may be superimposed on a video picture. The maximum TV display is 16 lines of 32 characters per line. The video rate is switchable between standard 525 line systems and the 1536 line CCTV system. Alternately, 128 character/line printout can be generated on the Videograph Printer. A cursor is available with 8 control commands.

1.11.4.5 Videograph Printer

The Videograph Printer is an electrostatic page-on-demand printer capable of delivering an 8 1/2 x 11 inch page every 800 milliseconds. Horizontal resolution is 128 lines per inch and vertical resolution is 1536 lines matching the slow rate, high resolution CCTV system. Gray scale resolution is four shades. The paper used is inexpensive zinc-oxide coated stock. When used with the Character Generator, the Videograph Printer serves as a high-speed page printer.

1.11.4.6 1536 F/S Cameras

The 1536 F/S Cameras are vidicon camera units which can be remotely selected to operate either in fast scan (15 frames per second) or slow scan (1.25 frames per second) mode. The format of both modes is 1536 lines per frame done in a sequential (non-interlace) scan. The aspect ratio is variable, but is set for a nominal 8 1/2 x 11 ratio. The camera bandwidth is limited to 9.5 Mhz for fast scan and 1.4 Mhz for slow scan.

1.11.4.7 Video Consoles

Each video console is a self-contained unit with two TV monitors and the necessary equipment for communicating with a digital computer. The video monitors consist of a 17 inch 1536 lines per frame slow (1.25 frames per second) monitor with a P-26 phosphor and a 17 inch 1536 lines per frame fast (15 frames per second) monitor with a VC-4 phosphor. Each monitor matches characteristics of the associated camera.

1.11.4.8 525 Line T.V. Cameras and Monitors

The 525 T.V. Cameras and Monitors are conventional television units; namely, 525 lines per frame, 30 frames per second interlaced (60 fields per second). These units provide for relatively low cost reduced resolution, which is sufficient for many message routing and simple acquisition and display purposes.

1.12 Intermachine Link

1.12.1 Function

The intermachine link consists of an interface between the PDP8/e Direct Memory Access Channel (single-cycle data break) and a 2701 Parallel Data Adapter attached to a Selector Channel on the IBM System 360/75. It is capable of half-duplex transfers at a rate of 500,000 bytes (8 bits) per second.

1.12.2 Documentation

Report

Borovec, R.T., "IMAGE 8: PDP8/e Link to IBM 2701 Parallel Data Adapter," Department of Computer Science Report, University of Illinois (in progress).

1.13 Low Speed Terminal Network

1.13.1 Function

The Low Speed Terminal Network is a console-oriented hardware configuration allowing users ready access to the operational components of Illiac III.

The hardware consists of a PDP8/e acting as a data concentrator for LINC tapes, Selectric terminals and teletypes.

An example of usage may be found in Report No. 429, "Show-And-Tell Systems Specifications," by John S. Read.

1.14 Power Distribution System

1.14.1 Function

The purpose of this system is to provide an effective means of controlling the power supplied to various subsystems of the Illiac III. The system allows individual sections of the machine to be de-energized for trouble-shooting and for initial calibration during checkout.

1.14.2 Design Features

The DC Power Distribution System comprises

1) the Primary Power Supplies
2) the Power Distribution Center, and finally
3) the Turn-on/Regulation Control.

This latter control can be subdivided into four stages: turn-on of the control voltage supplies, turn-on of the primary voltage supplies to 50% nominal ratings, regulation of the primary supplies at full load and finally regulation and remote sensing of the DC power as distributed to the individual sections of the machine.

The AC Power Distribution System provides two 200 amp secondary panels and a tertiary 20 amp service with distribution to the requisite sections of the machine.

1.14.3 Documentation

Manual

Martin, Ronald G. "Illiac III DC Power Distribution System", currently in draft form.

Krabbe, Paul, "Nomenclature, Hardware Location, and DC Power Distribution for the Illiac III Mainframe and Parts of the DC Power Distribution Wall", April 1970.

orm AEC-427
(6/68)
AECM 3201

U.S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

AEC REPORT NO. 433
COO-2118-0005

2. TITLE
ILLIAC III REFERENCE MANUAL VOLUME I:The Computer System

. TYPE OF DOCUMENT (Check one):

- [X] a. Scientific and technical report
- [ ] b. Conference paper not to be published in a journal:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [ ] c. Other (Specify) _____

RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- [X] a. AEC's normal announcement and distribution procedures may be followed.
- [ ] b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- [ ] c. Make no announcement or distrubution.

REASON FOR RECOMMENDED RESTRICTIONS:

SUBMITTED BY: NAME AND POSITION (Please print or type)

B. H. McCormick
Principal Investigator
Illiac III Project

Organization

University of Illinois
Department of Computer Science
Urbana, Illinois 61801

Signature Bruce H. McCormick

Date February 18, 1971

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

3. PATENT CLEARANCE:

- [ ] a. AEC patent clearance has been granted by responsible AEC patent group.
- [ ] b. Report has been sent to responsible AEC patent group for clearance.
- [ ] c. Patent clearance not required.